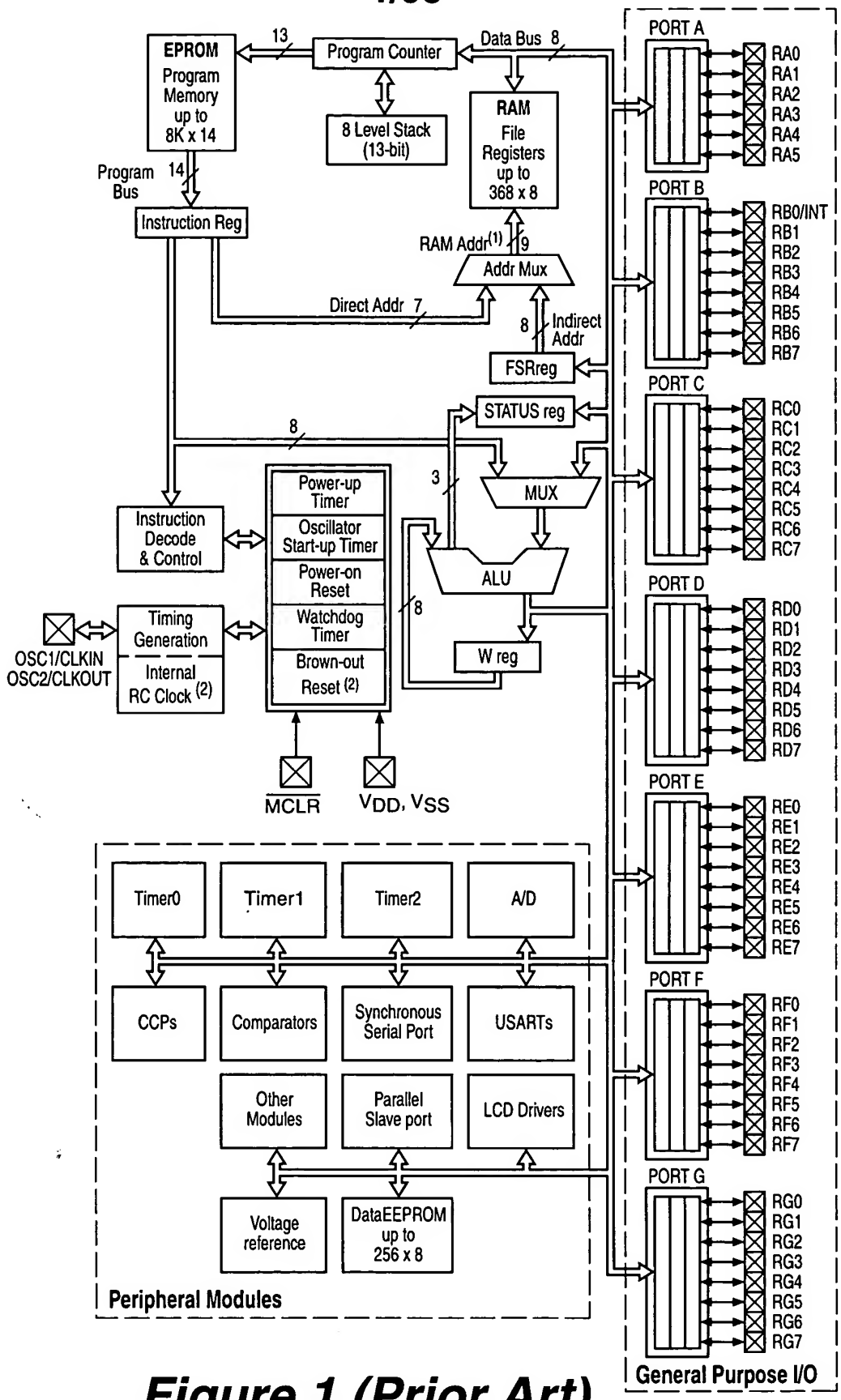


Figure 1 (Prior Art)

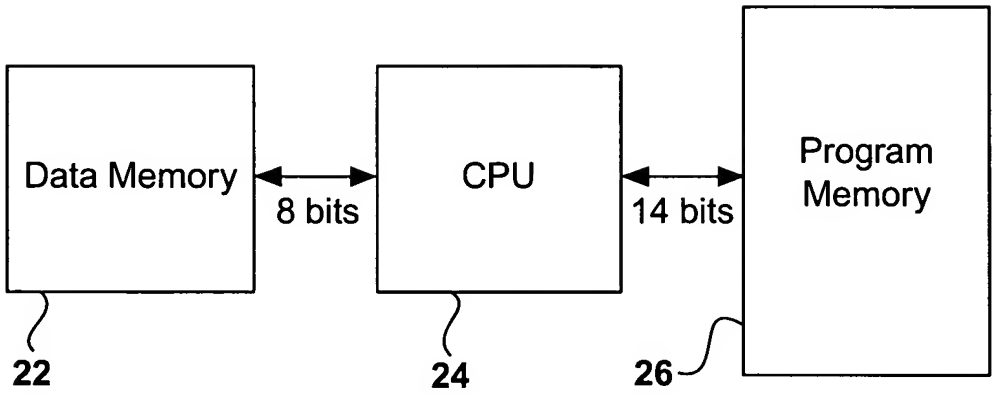


Figure 2 (Prior Art)

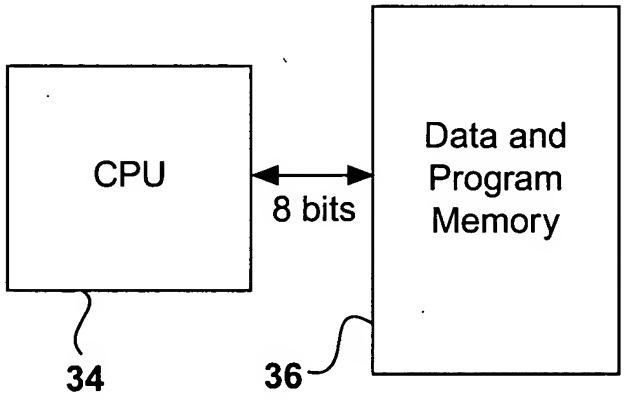


Figure 3 (Prior Art)

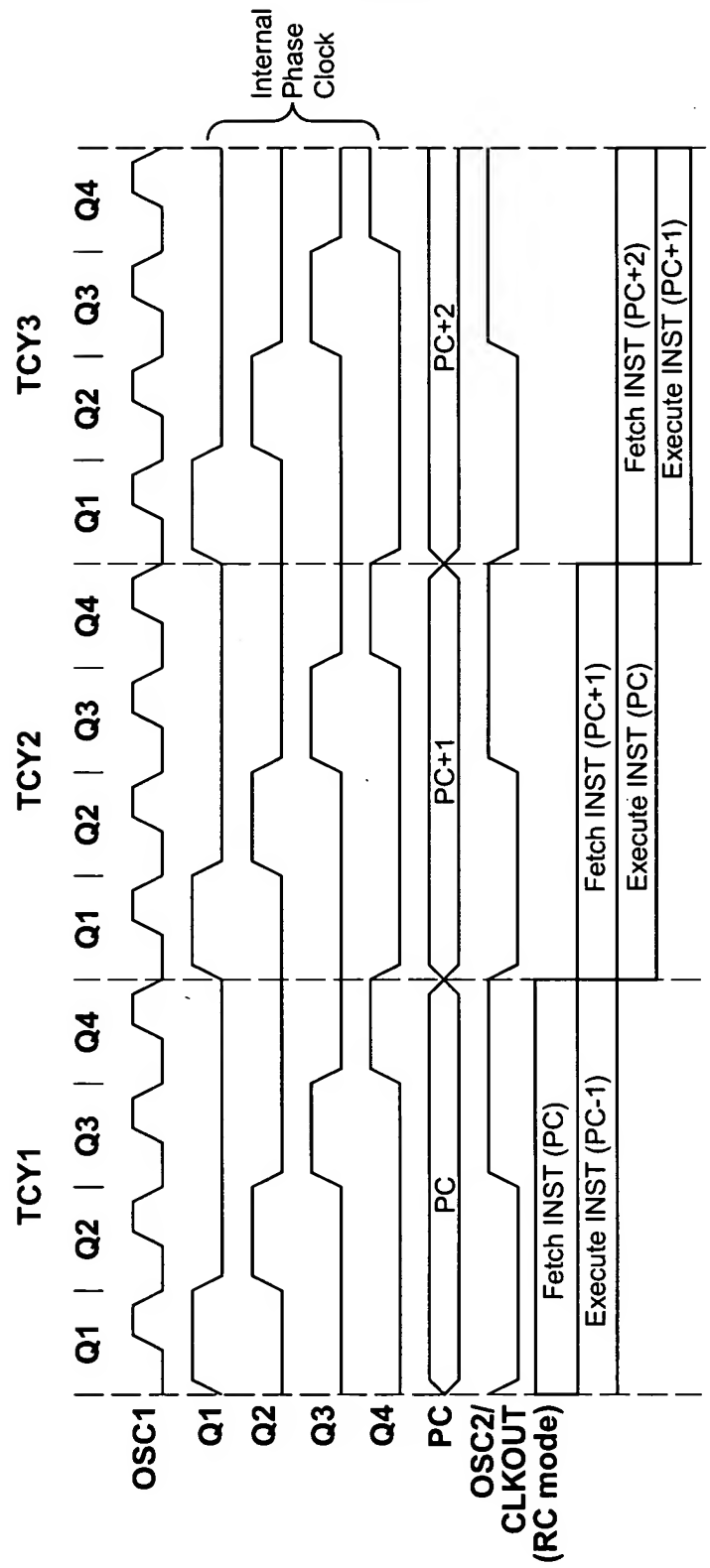


Figure 4 (Prior Art)

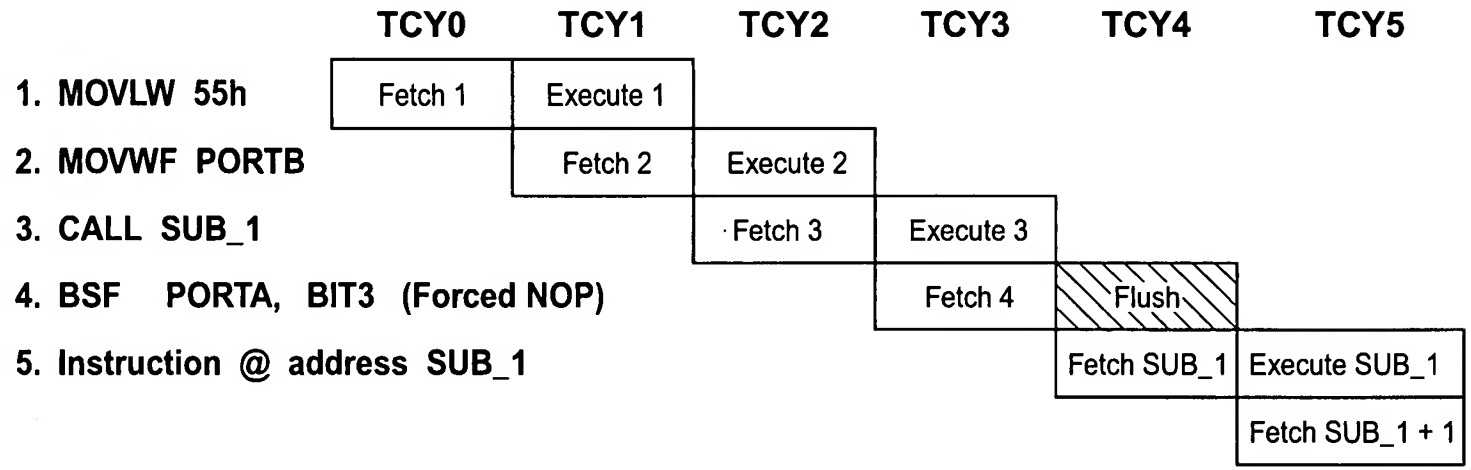


Figure 5 (Prior Art)

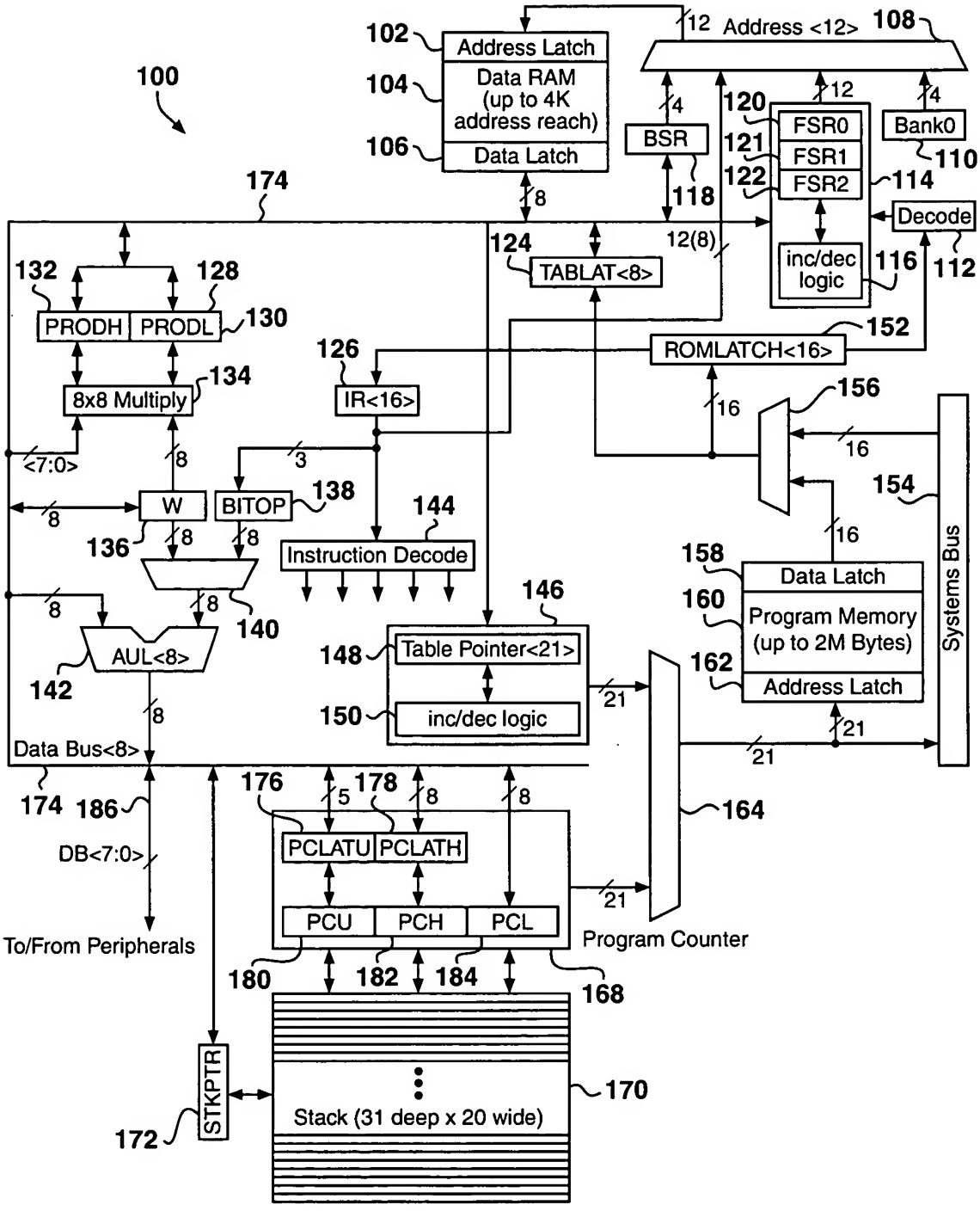

100
102
Address Latch
104
Data RAM
(up to 4K
address reach)
106
Data Latch
12
Address <12>
108
BSR
118
120
FSR0
121
FSR1
122
FSR2
12
4
Bank0
110
114
Decode
112
inc/dec
logic
116
174
124
TABLAT<8>
12(8)
132
128
PRODH
PRODL
130
126
IR<16>
ROMLATCH<16>
152
134
8x8 Multiply
16
156
<7:0>
8
3
16
154
W
BITOP
138
144
136
Instruction Decode
158
Data Latch
160
Program Memory
(up to 2M Bytes)
162
Address Latch
Systems Bus
140
146
AUL<8>
142
148
Table Pointer<21>
150
inc/dec logic
21
Data Bus<8>
174
186
DB<7:0>
To/From Peripherals
176
178
5
8
8
164
PCLATU
PCLATH
PCU
PCH
PCL
Program Counter
180
182
184
168
STKPTR
172
Stack (31 deep x 20 wide)
170

Figure 6

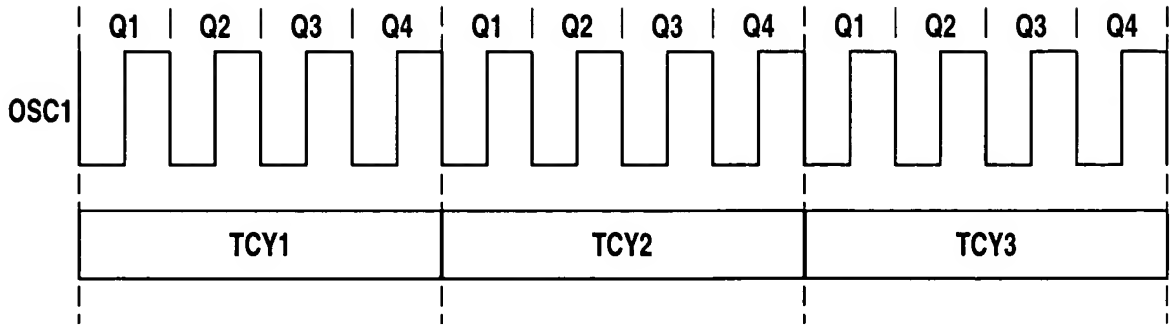


Figure 7

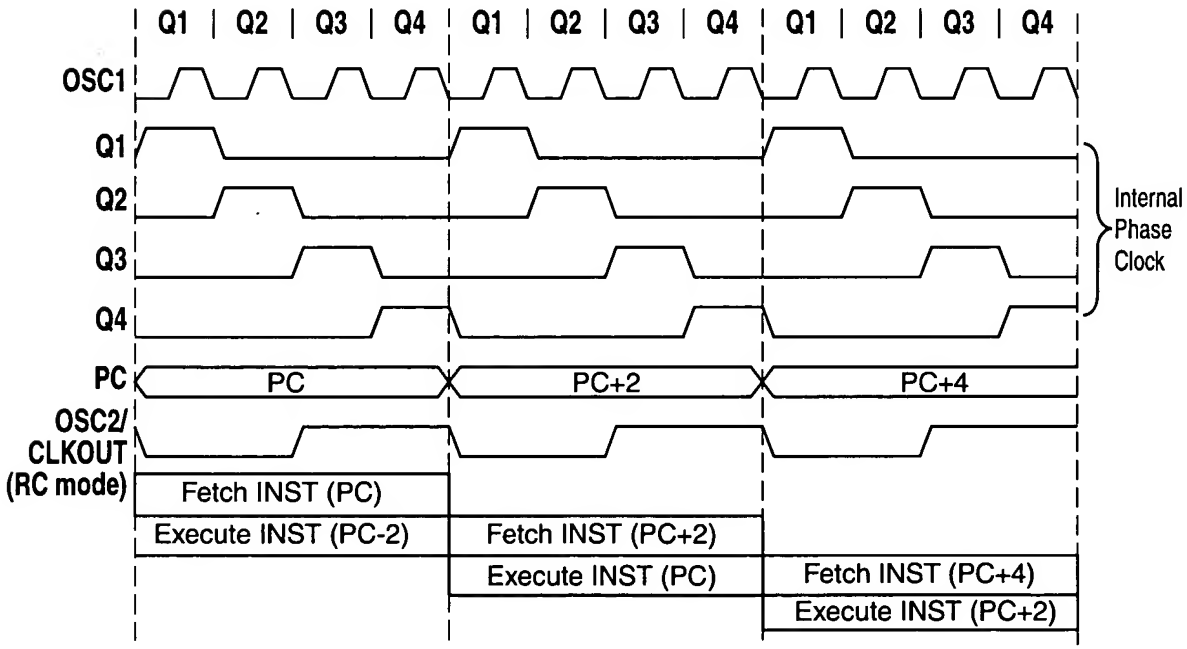


Figure 8

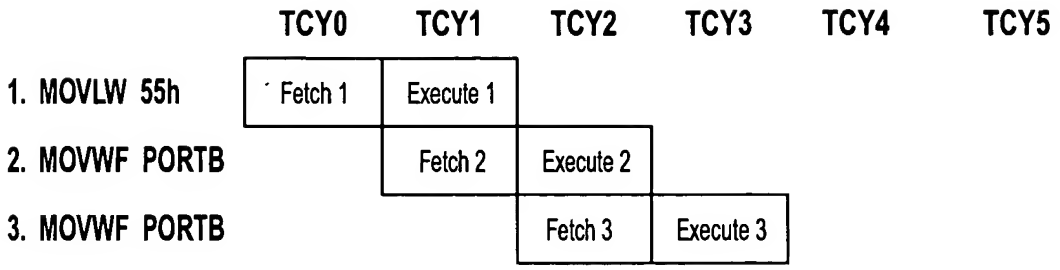


Figure 9

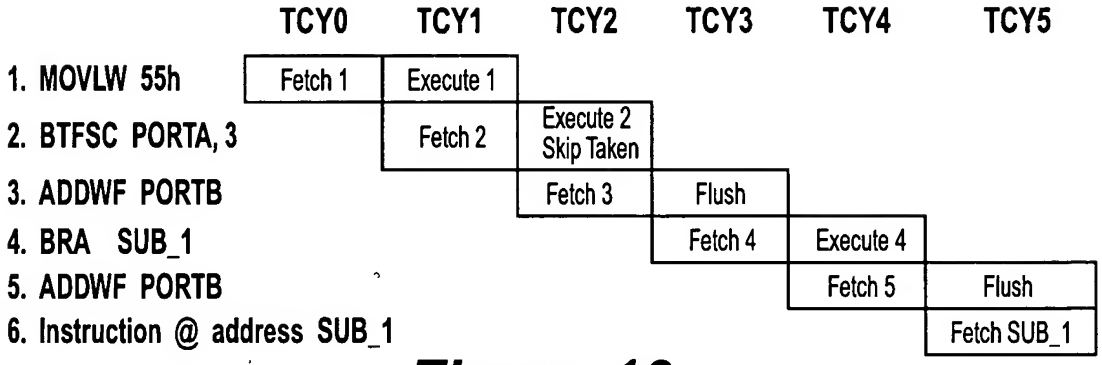


Figure 10

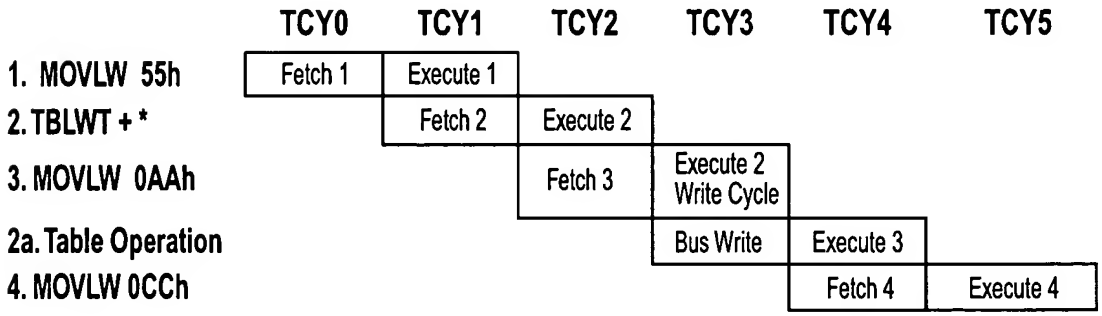


Figure 11

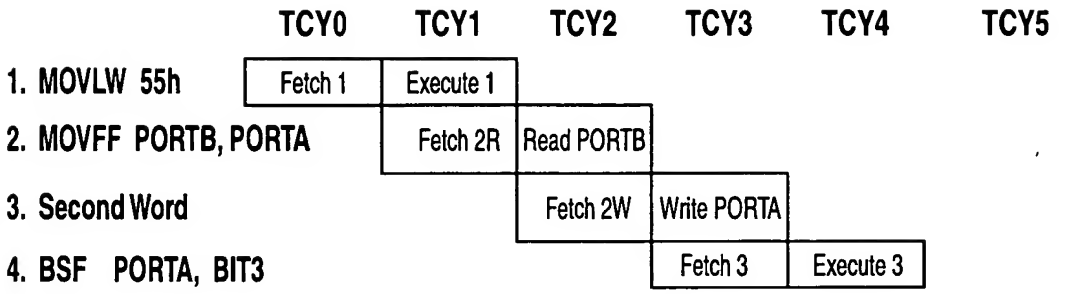


Figure 12

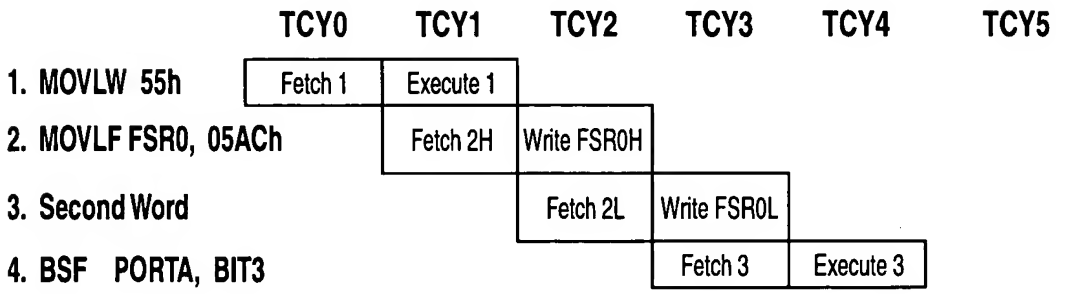


Figure 13

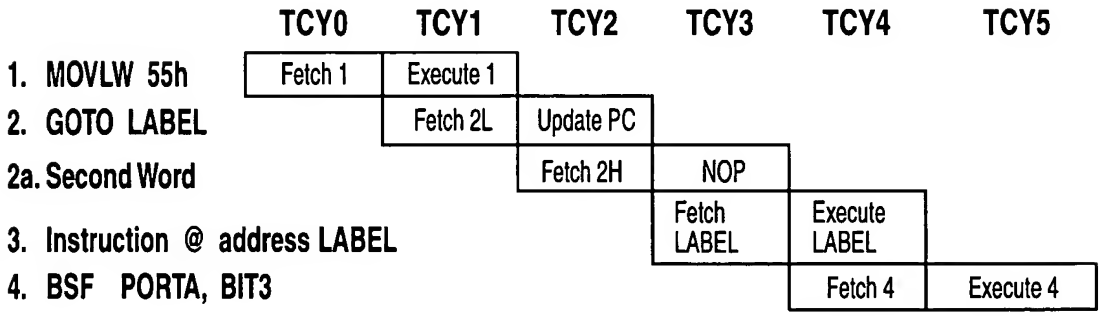


Figure 14

---	---	---	N	OV	Z	DC	C
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0

Figure 15

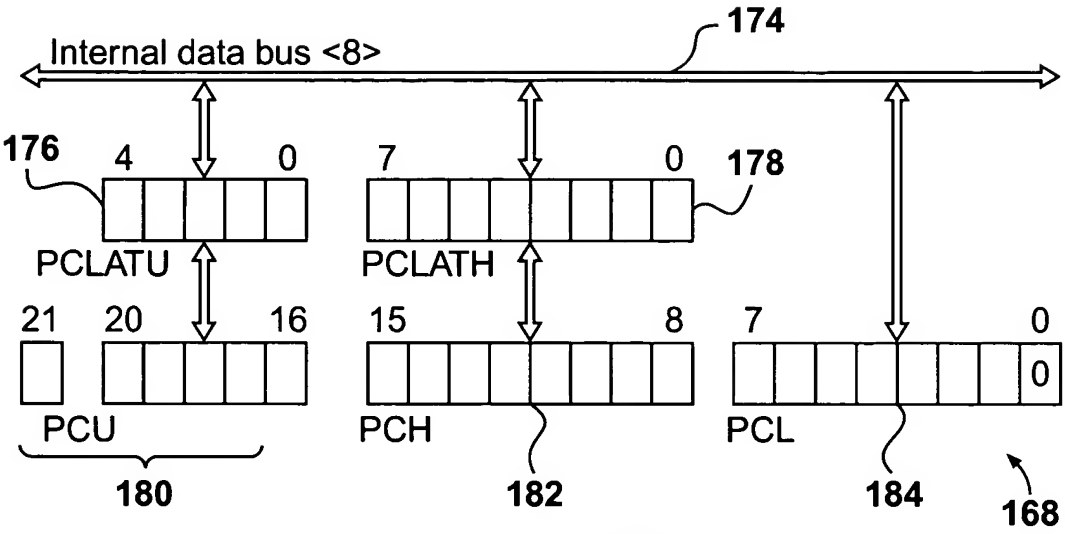


Figure 16

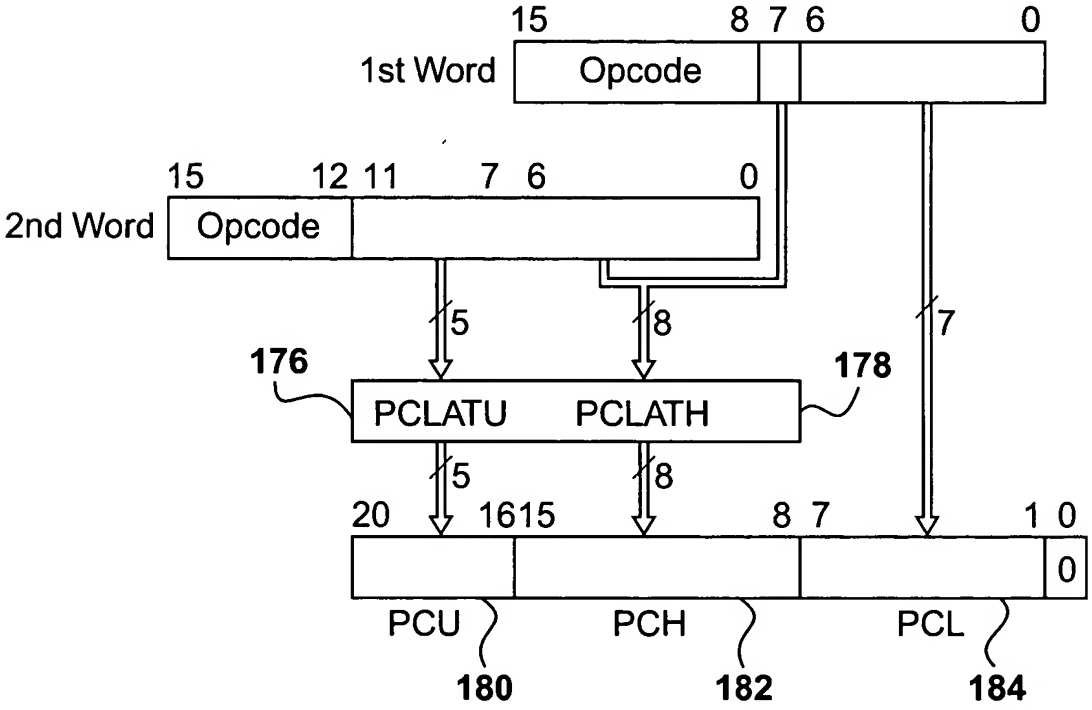


Figure 17

R/C-0	R/C-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STKOVF	STKUNF	-	SP4	SP3	SP2	SP1	SP0
bit7	6	5	4	3	2	1	bit0

R = Readable bit
W = Writeable bit
C = Clearable bit
U = Unimplemented bit, Read as '0'
- n = Value at POR reset

bit 7: STKOVF: Stack Overflow Flag bit
1 = Stack overflow occurred
0 = Reset or cleared by user software

bit 6: STKUNF: Stack Underflow Flag bit
1 = Stack underflow occurred
0 = Reset or cleared by user software

bit 5: Unimplemented: Read as '0'

bit 4-0: SP4:SP0: Stack Pointer Location bits

STKPTR - Stack Pointer Register

Figure 18

U-0	U-0	U-0	R/W-u	R/W-u	R/W-u	R/W-u	R/W-u
-	-	-	TOS20	TOS19	TOS18	TOS17	TOS16
bit7	6	5	4	3	2	1	bit0

R = Readable bit
W = Writeable bit
U = Unimplemented bit, Read as '0'
- n = Value at POR reset

bit 7-0: TSO<20:16>: Top of Stack bit

Figure 19 **TOSU - Top of Stack Upper**

R/W-u	R/W-u	R/W-u	R/W-u	R/W-u	R/W-u	R/W-u	R/W-u
TOS15	TOS14	TOS13	TOS12	TOS11	TOS10	TOS9	TOS8
bit7	6	5	4	3	2	1	bit0

R = Readable bit
W = Writeable bit
U = Unimplemented bit, Read as '0'
- n = Value at POR reset

bit 7-0: TOS<15:8>: Top of Stack bit

Figure 20 **TOSH - Top of Stack High**

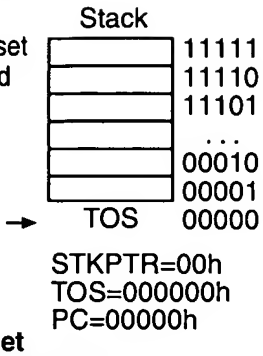
R/W-u	R/W-u	R/W-u	R/W-u	R/W-u	R/W-u	R/W-u	R/W-u
TOS7	TOS6	TOS5	TOS4	TOS3	TOS2	TOS1	TOS0
bit7	6	5	4	3	2	1	bit0

R = Readable bit
W = Writeable bit
U = Unimplemented bit, Read as '0'
- n = Value at POR reset

bit 7-0: TOS<7:0>: Top of Stack bit

Figure 21 **TOSL - Top of Stack Low**

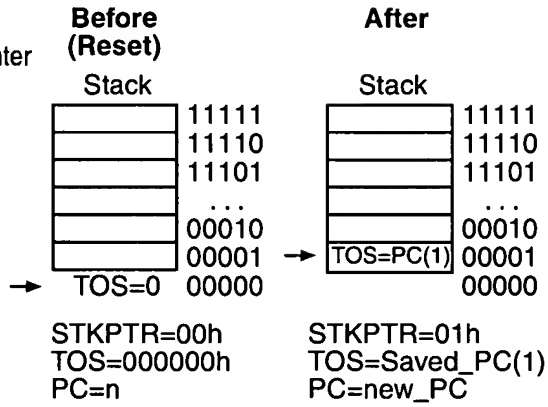
When a device is reset, the PC is loaded with the reset vector (0h). The stack pointer is initialized to 00h, and the Top of Stack register (TOS) is 000000h.



Reset

Figure 22

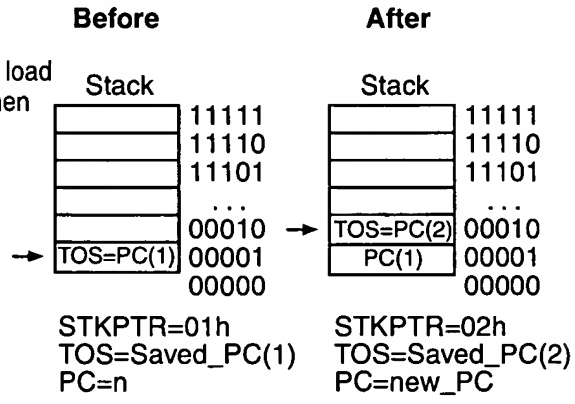
The first push of the stack increments the stack pointer to point to location 1. The value in the PC is loaded into stack level 1. The PC is then updated.



First CALL on an initialized stack

Figure 23

The second push will increment the stack pointer, load the TOS register with the current PC value, and then update the PC.

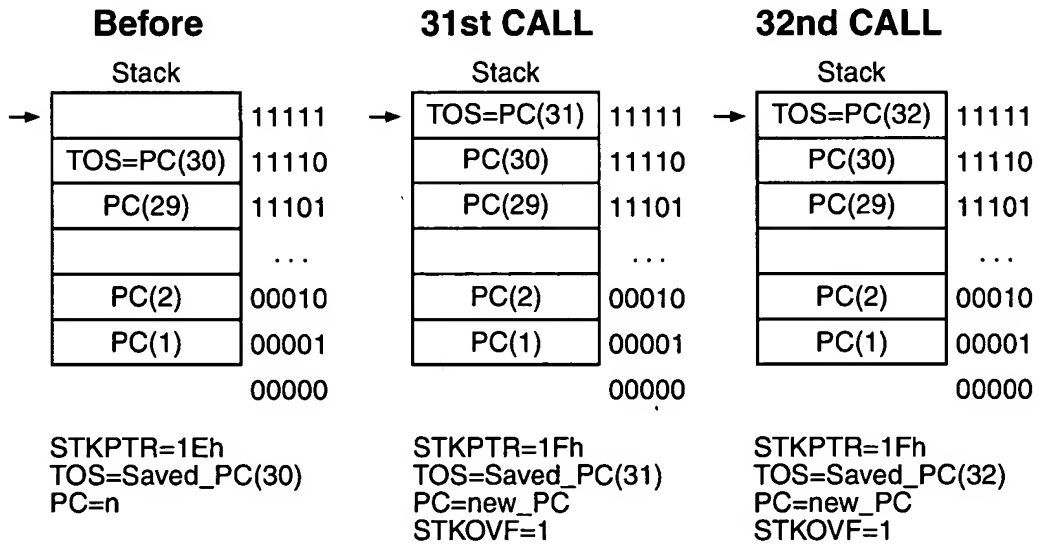


Second Consecutive CALL

Figure 24

The 31st CALL will increment the stack pointer, load the TOS register with the current PC value, and then update the PC. The STKOVF bit is set to indicate the impending stack overflow.

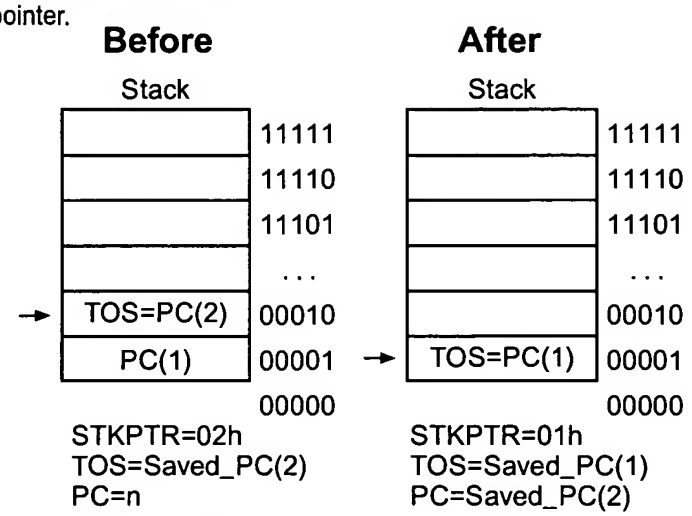
The 32nd CALL will attempt to increment the stack pointer. However the stack pointer is now pointing at the upper most stack level and cannot be incremented. The pointer will be loaded with 1Fh (still pointing to stack level 31), and the TOS will be overwritten with the current PC value. The PC will be updated. The stack overflow bit remains set. Another push will yield the same results. Once the pointer had incremented to 1Fh, it cannot be incremented to a higher value, it can only be cleared or decremented.



31st and 32nd consecutive CALL

Figure 25

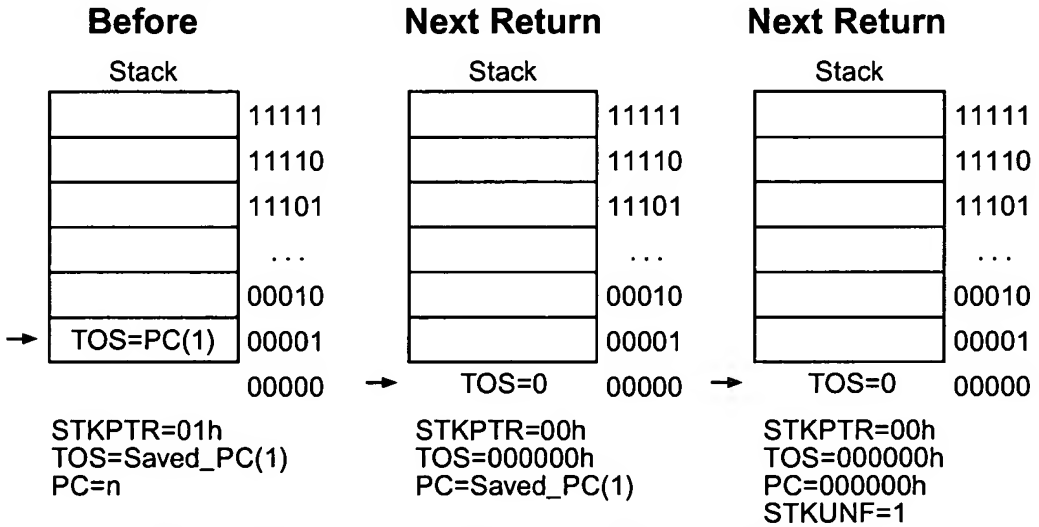
A return pop will copy the contents of the TOS to the PC and then decrement the stack pointer.



Return Pop

Figure 26

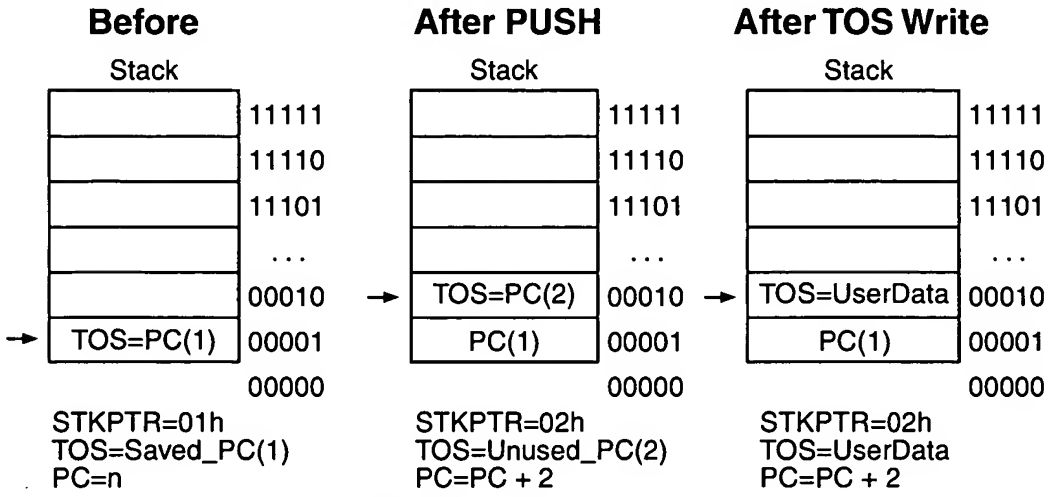
When the stack has been popped enough to reach 00h and the stack pointer can no longer be decremented, further popping will return 000000h to the PC. The stack pointer will maintain the value of 00h. The underflow bit (STKUNF) is set.



Stack Return Pops Causing Stack Underflow

Figure 27

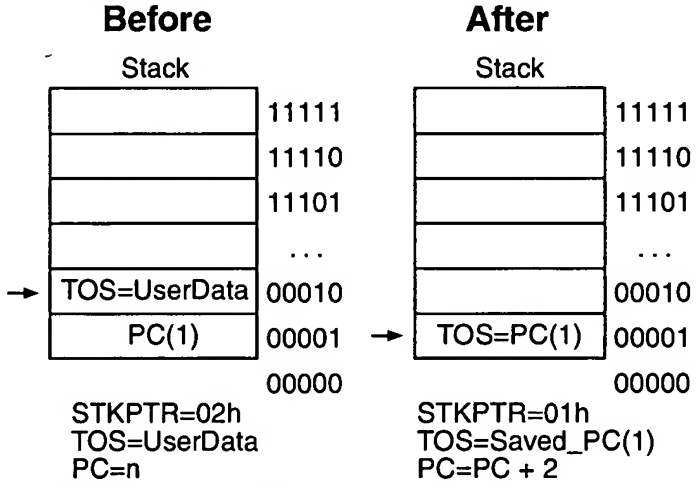
A PUSH instruction performs a similar operation as a call. The PC is incremented to PC+2, the stack pointer is incremented and the TOS is loaded with the PC value (which is essentially a wasted operation). The user will then have access to write values into the TOS registers.



PUSH instruction

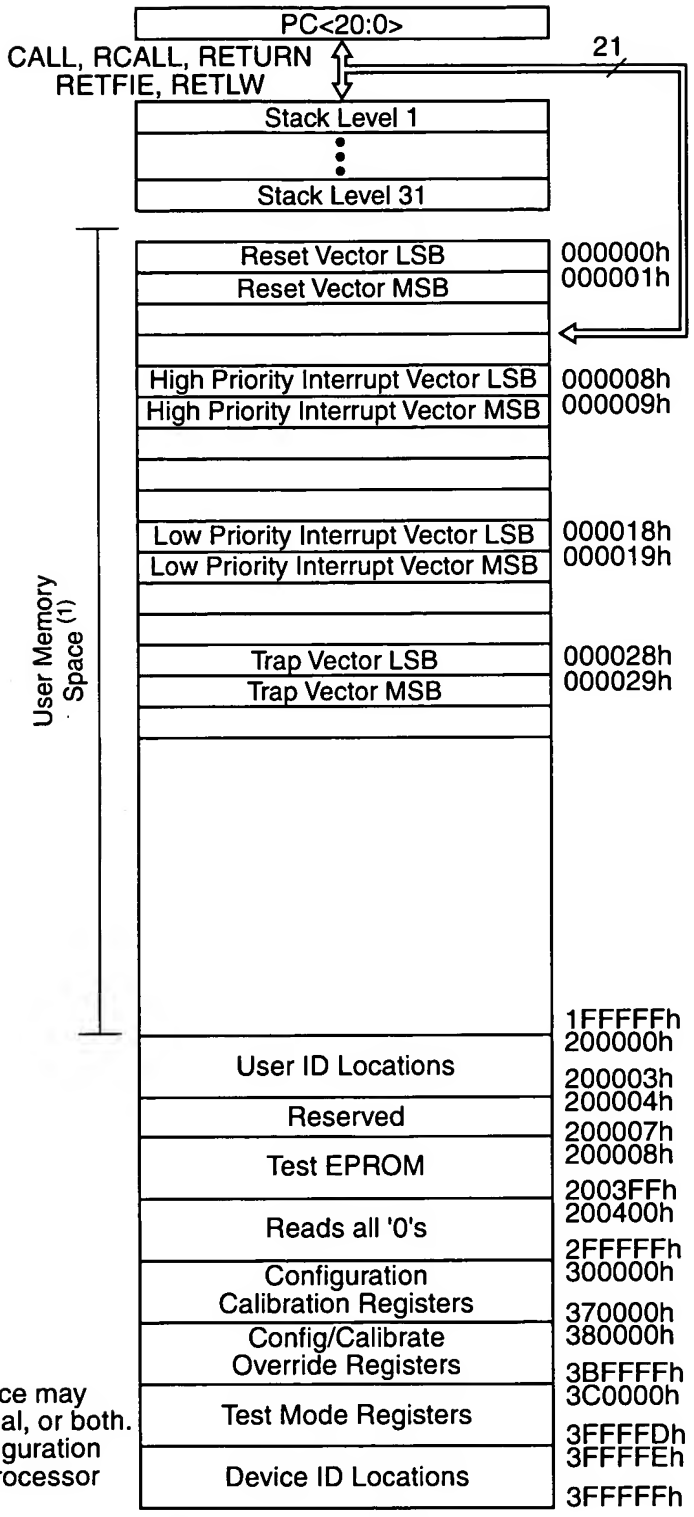
Figure 28

The POP instruction will perform actions similar to a return, however the PC is not loaded with the TOS value. The user must recover his data before the POP.
The stack pointer is then decremented.



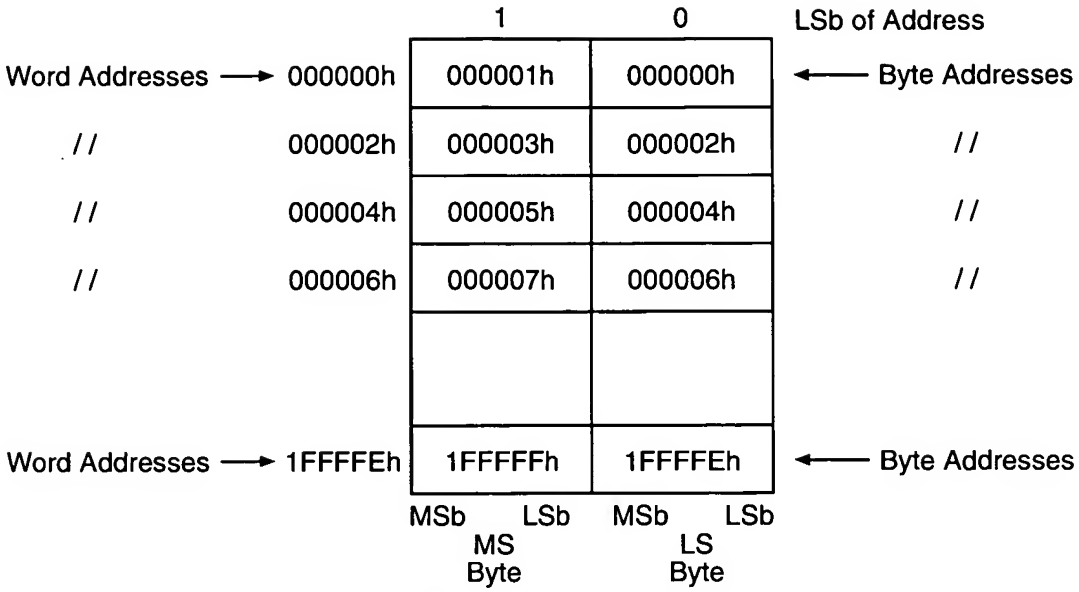
POP instruction

Figure 29



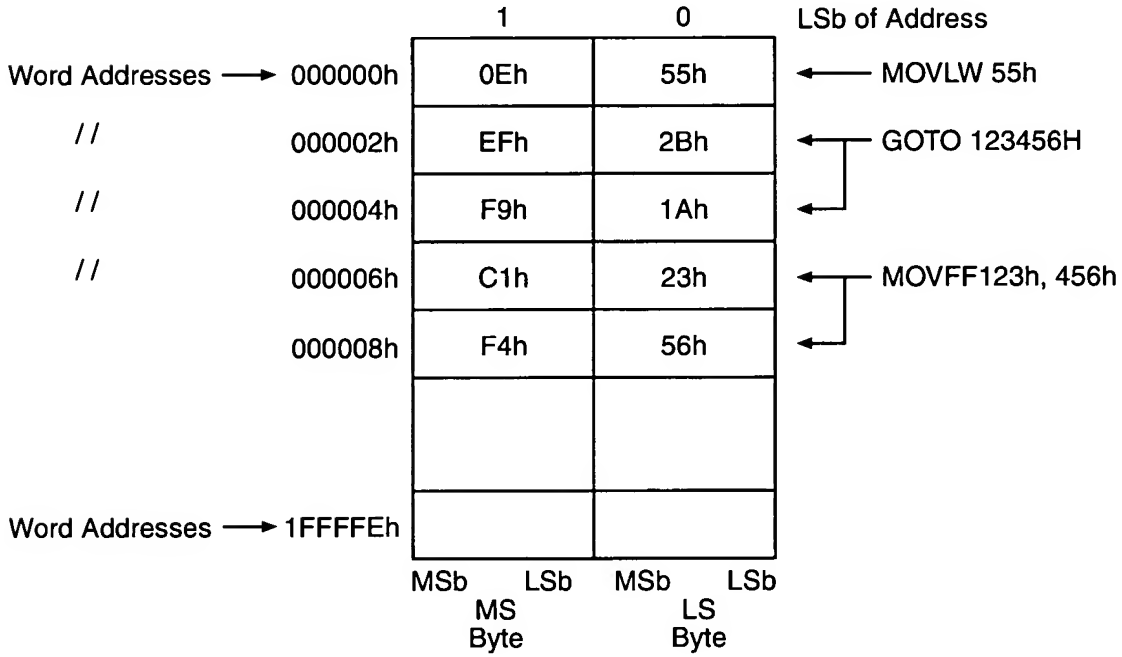
Note 1:
User memory space may be internal, external, or both. The memory configuration depends on the processor mode.

Figure 30



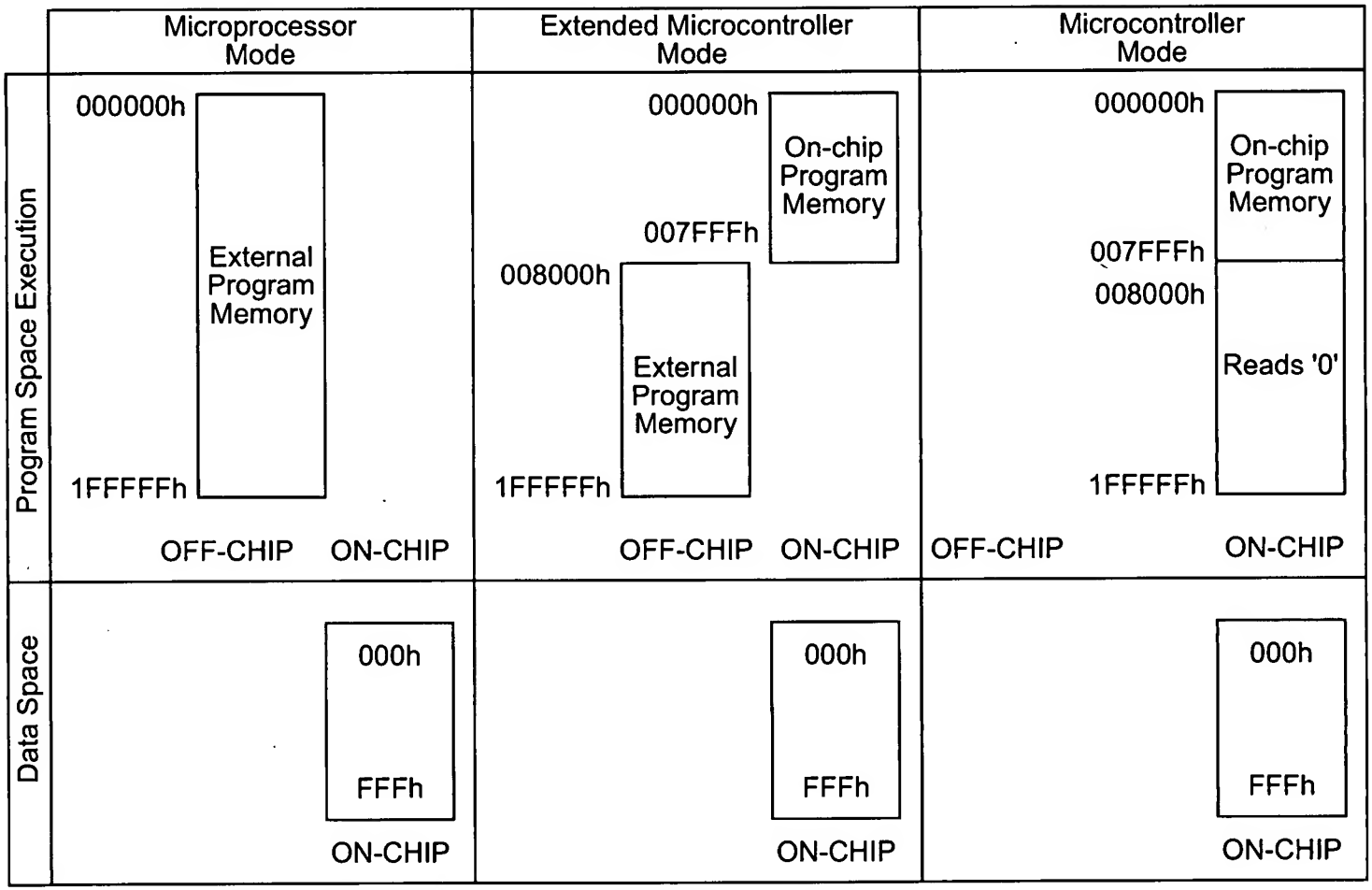
Memory Map in Bytes/Words

Figure 31



Instructions in Memory

Figure 32



Note 1: Example Shows 32K Bytes of Internal EPROM Program Memory

Device Memory Map in Different Modes

Figure 33

R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
EBDIS	—	WAIT1	WAIT0	—	—	WM1	WM0
bit7							bit0

R = Readable bit
W = Writable bit
-n = Value at POR reset
(x = unknown)

bit7: **EBDIS:** External bus disable
1 = Drivers disable, all external bus drivers disabled, I/0 ports can be used to control signals
0 = Drivers enabled

bit 6: **Unimplemented:** Read as '0'

bit 5-4: **Wait <1:0>:** Table reads and writes bus cycle wait count
11 = Table reads and writes will wait 0 Tcy
10 = Table reads and writes will wait 1 Tcy
01 = Table reads and writes will wait 2 Tcy
00 = Table reads and writes will wait 3 Tcy

bit 3-2: **Unimplemented:** Read as '0'

bit 1-0: **WM<1:0>:** TABLWT operation with 16-bit bus
1x = Word Write Mode: TABLAT0 and TABLAT1 word output, $\overline{WRH}$ active when TABLAT1 written
01 = Byte Select Mode: TABLAT data copied on both MS and LS, $\overline{WRH}$ and ($\overline{UB}$ or $\overline{LB}$) will activate
00 = Byte Write Mode: TABLAT data copied on both MS and Byte, $\overline{WRH}$ or $\overline{WRL}$ will activate

MEMCON Register

Figure 34

U-1	R/P-1	R/P-1	R/P-1	U-1	R/P-1	R/P-1	R/P-1
—	BSDIS	BADIS	WDIS	—	A19SDIS	A15DIS	A11DIS
bit7							bit0

R = Readable bit
P = Programmable bit
-n = UnprogrammedValue
(x = unknown)

bit 7: **Unimplemented:** Read as '0'

bit 6: **BSDIS:** Byte Select $\overline{UB}$, $\overline{LB}$ disable
0=Drivers disabled
1=Drivers enabled

bit 5: **BADIS:** Byte Address BA0 disable
0=Drivers disabled
1=Drivers enabled

bit 5: **WDIS:** Write Select $\overline{WRH}$, $\overline{WRL}$ disable
0=Drivers disabled
1=Drivers enabled

bit 3: **Unimplemented:** Read as '0'

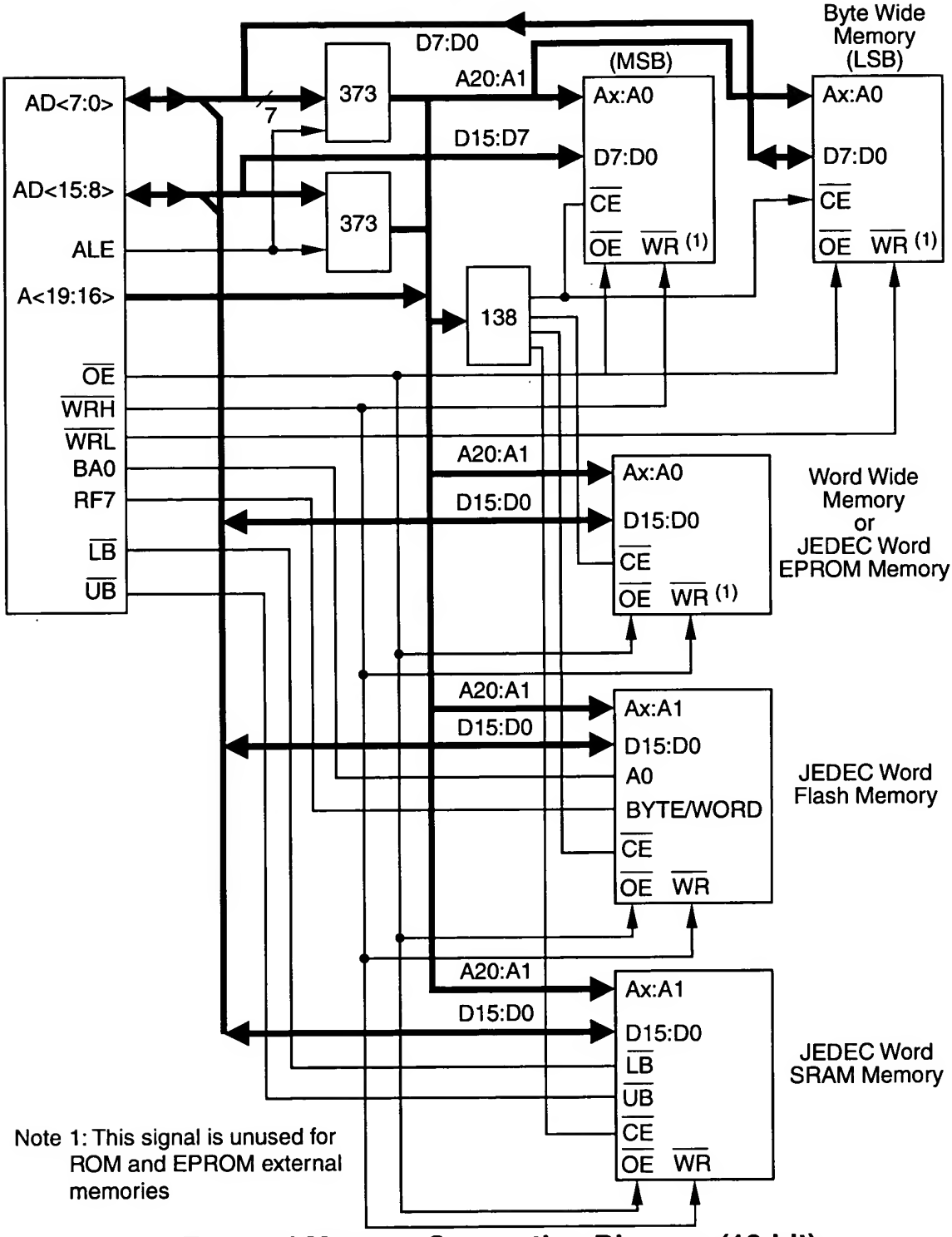
bit 2: **A19DIS:** Disable AD19:AD16 drivers
0=Drivers disabled
1=Drivers enabled

bit 1: **A15DIS:** Disable AD15:AD12 drivers
0=Drivers disabled, only if 8-bit external interface
1=Drivers enabled

bit 0: **A11DIS:** Disable AD11:AD8 drivers
0=Drivers disabled, only if 8-bit external interface
1=Drivers enabled

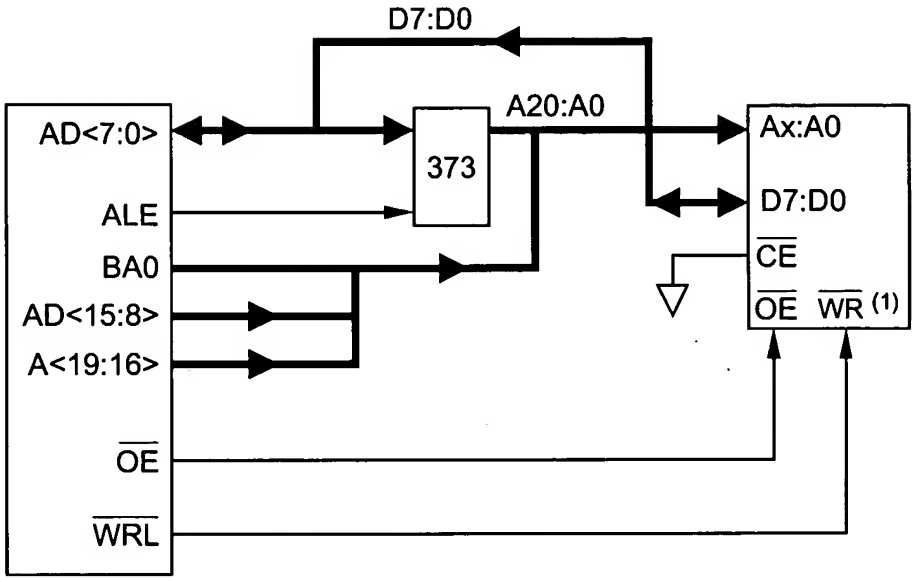
CONFIG7 Configuration Byte

Figure 35



External Memory Connection Diagram (16-bit)

Figure 36



Note 1: This signal is unused for ROM and EPROM external memories

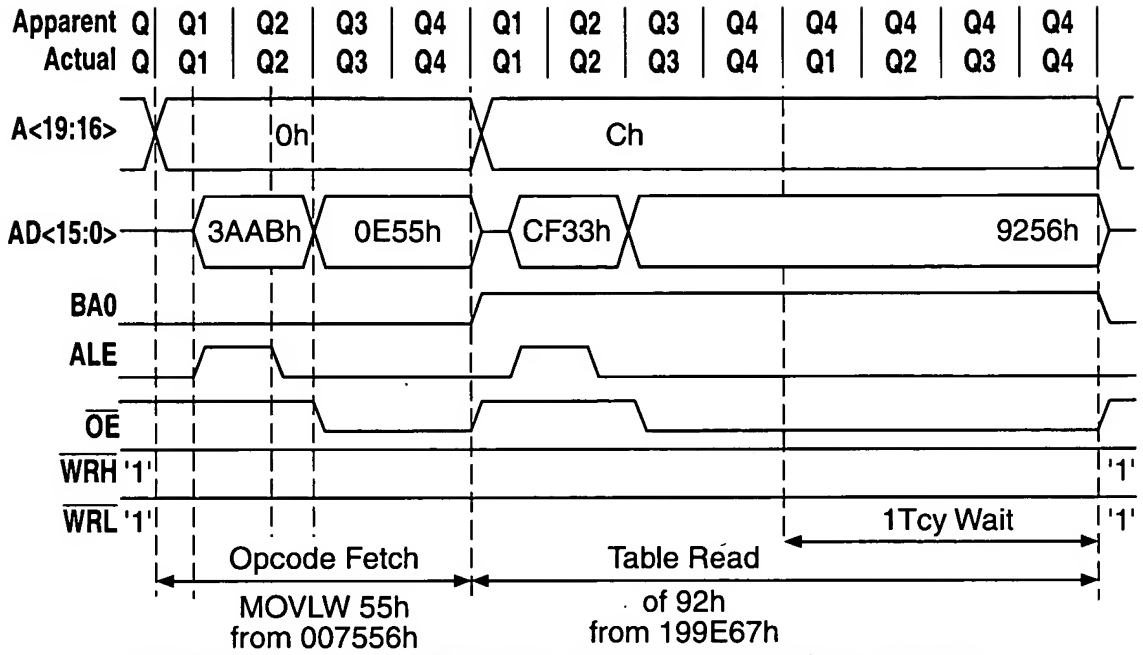
External Memory Connection Diagram (8-bit)

Figure 37

Name	Port	Bit	Function
RF4/BA0	PORTF	bit4	Input/Output or system bus byte address bit 0
RD0/AD0	PORTD	bit0	Input/Output or system bus address bit 0 or data bit 0
RD1/AD1	PORTD	bit1	Input/Output or system bus address bit 1 or data bit 1
RD2/AD2	PORTD	bit2	Input/Output or system bus address bit 2 or data bit 2
RD3/AD3	PORTD	bit3	Input/Output or system bus address bit 3 or data bit 3
RD4/AD4	PORTD	bit4	Input/Output or system bus address bit 4 or data bit 4
RD5/AD5	PORTD	bit5	Input/Output or system bus address bit 5 or data bit 5
RD6/AD6	PORTD	bit6	Input/Output or system bus address bit 6 or data bit 6
RD7/AD7	PORTD	bit7	Input/Output or system bus address bit 7 or data bit 7
RE0/AD8	PORTE	bit0	Input/Output or system bus address bit 8 or data bit 8
RE1/AD9	PORTE	bit1	Input/Output or system bus address bit 9 or data bit 9
RE2/AD10	PORTE	bit2	Input/Output or system bus address bit 10 or data bit 10
RE3/AD11	PORTE	bit3	Input/Output or system bus address bit 11 or data bit 11
RE4/AD12	PORTE	bit4	Input/Output or system bus address bit 12 or data bit 12
RE5/AD13	PORTE	bit5	Input/Output or system bus address bit 13 or data bit 13
RE6/AD14	PORTE	bit6	Input/Output or system bus address bit 14 or data bit 14
RE7/AD15	PORTE	bit7	Input/Output or system bus address bit 15 or data bit 15
RG0/A16	PORTG	bit0	Input/Output or system bus address bit 16
RG1/A17	PORTG	bit1	Input/Output or system bus address bit 17
RG2/A18	PORTG	bit2	Input/Output or system bus address bit 18
RG3/A19	PORTG	bit3	Input/Output or system bus address bit 19
RF0/ALE	PORTF	bit0	Input/Output or system bus Address Latch Enable (ALE) control pin
RF1/$\overline{OE}$	PORTF	bit1	Input/Output or systems bus Output Enable ($\overline{OE}$) control pin
RF2/$\overline{WRL}$	PORTF	bit2	Input/Output or system bus Write Low ($\overline{WRL}$) control pin
RF3/$\overline{WRH}$	PORTF	bit3	Input/Output or system bus Write High ($\overline{WRH}$) control pin
RF5/$\overline{LB}$	PORTF	bit2	Input/Output or system bus Lower Byte Enable ($\overline{LB}$) control pin
RF6/$\overline{UB}$	PORTF	bit3	Input/Output or system bus Upper Byte Enable ($\overline{UB}$) control pin

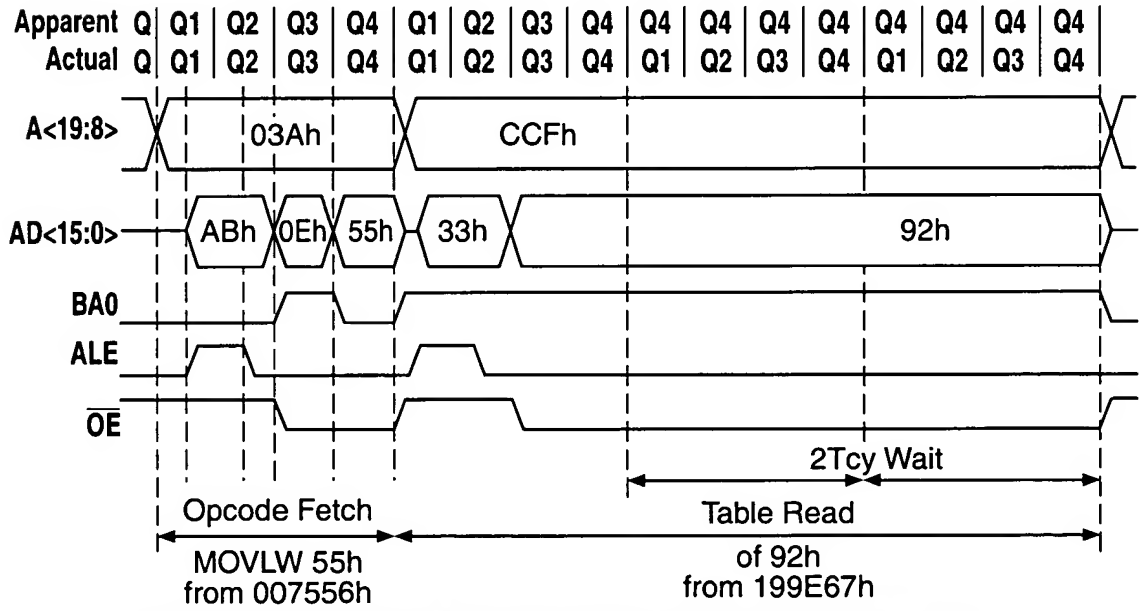
Typical Port Functions

Figure 38



External Program Memory Bus Timing (16-bit Mode)

Figure 39



External Program Memory Bus Timing (8-bit Mode)

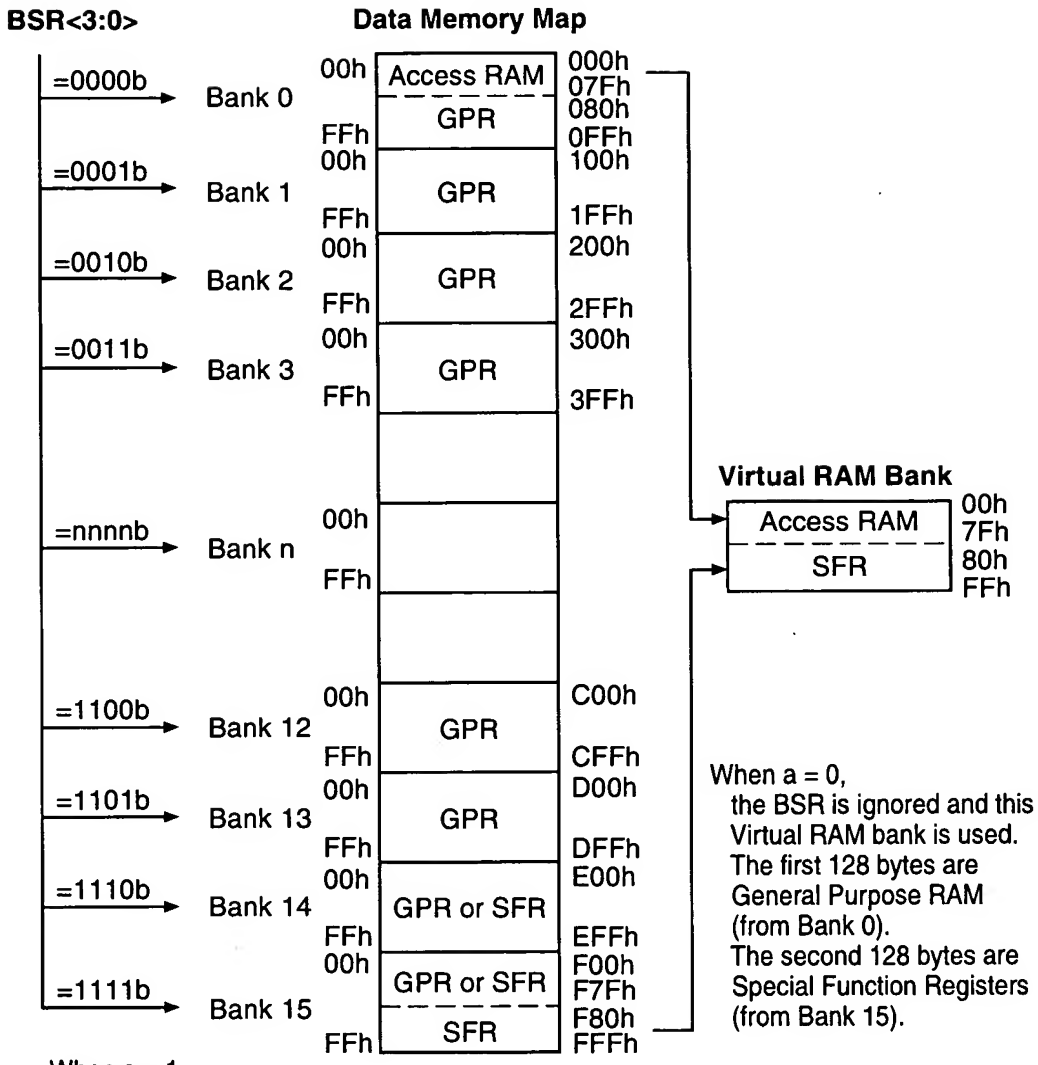
Figure 40

	A<19:16>	AD<15:8>	AD<7:0>	BA0	$\overline{ALE}$	$\overline{OE}$	$\overline{WRH}$	$\overline{WRL}$	$\overline{UB}$	$\overline{UL}$
Opcode Fetch 8-bit	PC<20:17>	PC<16:9>	Q1-2:PC<8:1> Q3: INST<15:8> Q4: INST<7:0>	Q1-2:0 Q3: 1 Q4: 0	1	0	-	1	-	-
Opcode Fetch 16-bit	PC<20:17>	Q2:PC<20:17> Q3-4: INST<15:8>	Q1-2:PC<20:17> Q3-4: INST<7:0>	0	1	0	1	1	0	0
Table Read 8-bit	TBLPTR<20:17>	TBLPTR<16:9>	Q1-2:TBLPTR<8:1> Q3-4:DATA<7:0>	TBLPTR<0>	1	0	-	1	-	-
Table Read 16-bit	TBLPTR<20:17>	Q1-2:TBLPTR<16:9> Q3-4:DATA<15:8>	Q1-2:TBLPTR<8:1> Q3-4:DATA<7:0>	TBLPTR<0>	1	0	1	1	0	0
Table Write 8-bit	TBLPTR<20:17>	TBLPTR<16:9>	Q1-2:TBLPTR<8:1> Q3-4:TABLAT<7:0>	TBLPTR<0>	1	1	-	0	-	-
Table Write 16-bit Byte Write Mode	TBLPTR<20:17>	Q1-2:TBLPTR<16:9> Q3-4:TABLAT<7:0>	Q1-2:TBLPTR<8:1> Q3-4:TABLAT<7:0>	TBLPTR<0>	1	1	$\overline{XH}$	$\overline{XL}$	1	1
Table Write16-bit Byte Select Mode	TBLPTR<20:17>	Q1-2:TBLPTR<16:9> Q3-4:TABLAT<7:0>	Q1-2:TBLPTR<8:1> Q3-4:TABLAT<7:0>	TBLPTR<0>	1	1	0	1	$\overline{YH}$	$\overline{YL}$
Table Write16-bit Word Write Mode TABPTR<0>=0	TBLPTR<20:17>	Q1-2:TBLPTR<16:9> Q3-4: Hi-Z	Q1-2:TBLPTR<8:1> Q3-4: Hi-Z TBHREG<7:0>= TABLAT<7:0>	TBLPTR<0> =0	1	1	1	1	1	1
Table Write16-bit Word Write Mode TABPTR<0>=1	TBLPTR<20:17>	Q1-2:TBLPTR<16:9> Q3-4:TABLAT<7:0>	Q1-2:TBLPTR<8:1> Q3-4:TBHREG<7:0>	TBLPTR<0> =1	1	1	0	1	0	0

TBHREG is TABLAT high byte holding register
XH= WRITE signal .AND. $\overline{TBLPTR<0>}$
XL= WRITE signal .AND. $\overline{TBLPTR<0>}$
YH= TBLPTR<0>
YL= $\overline{TBLPTR<0>}$

External Bus Cycle Types

Figure 41



The Data Memory Map and the Instruction 'a' bit

Figure 42

FFFh	TOSU	FDFh	INDF2	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2	FBCh	CCPR2H	F9Ch	MEMCON
FFBh	PCLATU	FDBh	PLUSW2	FBBh	CCPR2L	F9Bh	
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	DDRJ
FF9h	PCL	FD9h	FSR2L	FB9h	CCPR3H	F99h	DDRH
FF8h	TBLPTRU	FD8h	STATUS	FB8h	CCPR3L	F98h	DDRG
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	CCP3CON	F97h	DDRF
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	CCPR4H	F96h	DDRE
FF5h	TABLAT	FD5h	T0CON	FB5h	CCPR4L	F95h	DDRD
FF4h	PRODH	FD4h	rsvd DEBUG	FB4h	CCP4CON	F94h	DDRC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	DDRB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	DDRA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	LATJ
FF0h	INTCON3	FD0h	RCON	FB0h		F90h	LATH
FEFh	INDF0	FCFh	TMR1H	FAFh	COM1BRG	F8Fh	LATG
FEEh	POSTINC0	FCEh	TMR1L	FAEh	COM1REC	F8Eh	LATF
FEDh	POSTDEC0	FCDh	T1CON	FADh	COM1TX	F8Dh	LATE
FECh	PREINC0	FCCh	TMR2	FACh	COM1STA	F8Ch	LATD
FEBh	PLUSW0	FCBh	PR2	FABh	COM1CON	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	COM2BRG	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	COM2REC	F89h	LATA
FE8h	W	FC8h	SSPADD	FA8h	COM2TX	F88h	PORTJ
FE7h	INDF1	FC7h	SSPSTAT	FA7h	COM2STA	F87h	PORTH
FE6h	POSTINC1	FC6h	SSPCON1	FA6h	COM2CON	F86h	PORTG
FE5h	POSTDEC1	FC5h	SSPCON2	FA5h	IPR3	F85h	PORTF
FE4h	PREINC1	FC4h	ADRESH	FA4h	PIR3	F84h	PORTE
FE3h	PLUSW1	FC3h	ADRESL	FA3h	PIE3	F83h	PORTD
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA

Special Function Register Map

Figure 43

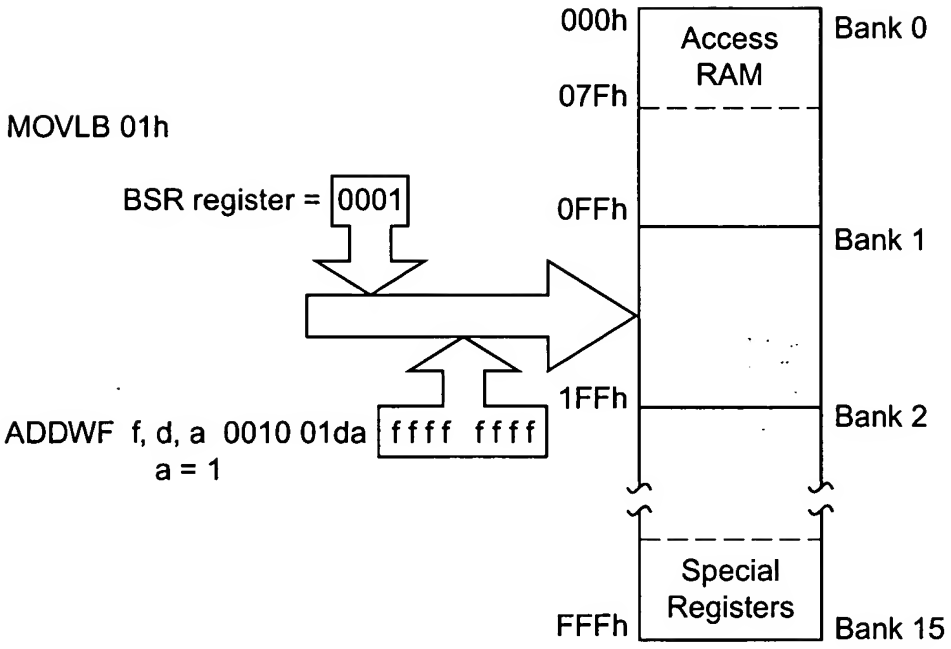
Filename		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other resets (note 3)
FFh	TOSU	-	-	-	Top-of-Stack upper Byte (TOS<20:16>)					- - -0 0000	- - -0 0000
FEh	TOSH	Top-of-Stack High Byte (TOS<15:8>)								0000 0000	0000 0000
FDh	TOSL	Top-of-Stack Low Byte (TOS<7:0>)								0000 0000	0000 0000
FCh	STKPTR	STKOVF	STKUNF	-	Return Stack Pointer					00 - 0 0000	00 - 0 0000
FBh	PCLATU	-	-	Holding Register for PC<21:16>						- - 00 0000	- - 00 0000
FAh	PCLATH	Holding Register for PC<15:8>								0000 0000	0000 0000
F9h	PCL	PC Low Byte (PC<7:0>								0000 0000	0000 0000
F8h	TBLPTRU	-	-	Program Memory Table Pointer Upper Byte (TBLPTR<21:16>)						- - 00 0000	- - 00 0000
F7h	TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								0000 0000	0000 0000
F6h	TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								0000 0000	0000 0000
F5h	TABLAT	Program Memory Table Latch								0000 0000	0000 0000
F4h	PRODH	Product Register High Byte								xxxx xxxx	uuuu uuuu
F3h	PRODL	Product Register Low Byte								xxxx xxxx	uuuu uuuu
F2h	INTCON	GIE/GIEH	PEIE/GIEL	T0IE	INT0E	RBIE	T0IF	INT0F	RBIF	0000 000x	0000 000x
F1h	INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	T0IP	INT3P	RPIP	1111 1111	1111 1111
F0h	INTCON3	INT2P	INT1P	INT3E	INT2E	INT1E	INT3F	INT2F	INT1F	1100 0000	1100 0000
EFh	INDF0	Uses contents of FSR0 to address data memory - value of FSR0 not changed (not a physical register)								n/a	n/a
EEh	POSTINC0	Uses contents of FSR0 to address data memory - value of FSR0 post-increment (not a physical register)								n/a	n/a
EDh	POSTDEC0	Uses contents of FSR0 to address data memory - value of FSR0 post-decrement (not a physical register)								n/a	n/a
ECh	PREINC0	Uses contents of FSR0 to address data memory - value of FSR0 pre-incremented (not a physical register)								n/a	n/a
EBh	PLUSW0	Uses contents of FSR0 to address data memory - value of FSR0 offset by W (not a physical register)								n/a	n/a
EAh	FSR0H	-	-	-	-	Indirect Data Memory Address Pointer 0 High				- - - - xxxx	- - - - uuuu
E9h	FSR0L	Indirect Data Memory Address Pointer 0 Low Byte								xxxx xxxx	uuuu uuu
E8h	W	Working Register								xxx xxxx	uuuu uuuu
E7h	INDF1	Uses contents of FSR1 to address data memory - value of FSR1 not changed (not a physical register)								n/a	n/a
E6h	POSTINC1	Uses contents of FSR1 to address data memory - value of FSR1 post-increment (not a physical register)								n/a	n/a
E5h	POSTDEC1	Uses contents of FSR1 to address data memory - value of FSR1 post-decrement (not a physical register)								n/a	n/a
E4h	PREINC1	Uses contents of FSR1 to address data memory - value of FSR1 pre-incremented (not a physical register)								n/a	n/a
E3h	PLUSW1	Uses contents of FSR1 to address data memory - value of FSR1 offset by W (not a physical register)								n/a	n/a
E2h	FSR1H	-	-	-	-	Indirect Data Memory Address Pointer 1 High				- - - - xxxx	- - - - uuuu
E1h	FSR1L	Indirect Data Memory Address Pointer 1 Low Byte								xxxx xxxx	uuuu uuuu
E0h	BSR	-	-	-	-	Bank Select Register				- - - - 0000	- - - - 0000
DFh	INDF2	Uses contents of FSR2 to address data memory - value of FSR2 not changed (not a physical register)								n/a	n/a
DEh	POSTINC2	Uses contents of FSR2 to address data memory - value of FSR2 post-increment (not a physical register)								n/a	n/a
DDh	POSTDEC2	Uses contents of FSR2 to address data memory - value of FSR2 post-decrement (not a physical register)								n/a	n/a
DCh	PREINC2	Uses contents of FSR2 to address data memory - value of FSR2 pre-incremented (not a physical register)								n/a	n/a
DBh	PLUSW2	Uses contents of FSR2 to address data memory - value of FSR2 offset by W (not a physical register)								n/a	n/a
DAh	FSR2H	-	-	-	-	Indirect Data Memory Address Pointer 2 High				- - - - xxxx	- - - - uuuu
D9h	FSR2L	Indirect Data Memory Address Pointer 2 Low Byte								xxxx xxxx	uuuu uuuu
D8h	STATUS	-	-	-	N	OV	Z	DC	C	- - - x xxxx	- - - u uuuu

Core Special Function Register Summary

Figure 44

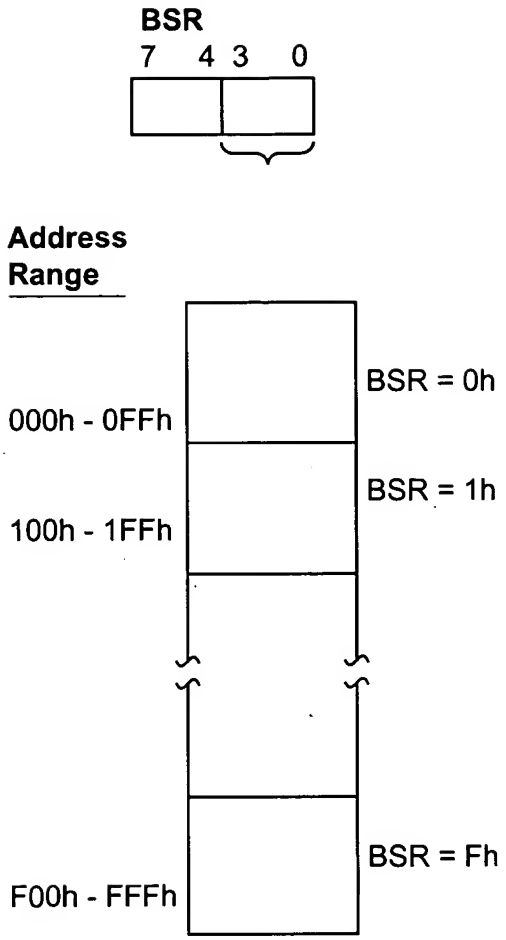
Filename		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other resets (note 3)
D3h	OSCCON	-	-	-	-	-	-	-	SCS	---- ---0	---- ---0
D2h	LVDCON	-	-	BGST	LVDEN	LVV3	LVV2	LVV1	LVV0	--00 0101	--00 0101
D1h	WDTCON	-	-	-	-	-	-	-	SWDTE	---- ---0	---- ---0
D0h	RCON	IPE	LWRT	-	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	00-1 11qq	00-q qquu
A5h	IPR3	-	-	-	-	-	-	-	-	---- ----	---- ----
A4h	PIR3	-	-	-	-	-	-	-	-	---- ----	---- ----
A3h	PIE3	-	-	-	-	-	-	-	-	---- ----	---- ----
A2h	IPR2	-	-	-	-	BCLIP	LVDIP	TMR3IP	CCP2IP	--- 1111	--- 1111
A1h	PIR2	-	-	-	-	BCLIF	LVDIF	TMR3IF	CCP2IF	--- 0000	--- 0000
A0h	PIE2	-	-	-	-	BCLIE	LVDIE	TMR3IE	CCP2IE	---- 0000	--- 0000
9Fh	IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
9Eh	PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
9Dh	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
9Ch	MEMCON	EBDIS	-	WAIT1	WAIT0	-	-	WM1	WM0	0-00 --00	0-00 --00

Figure 45



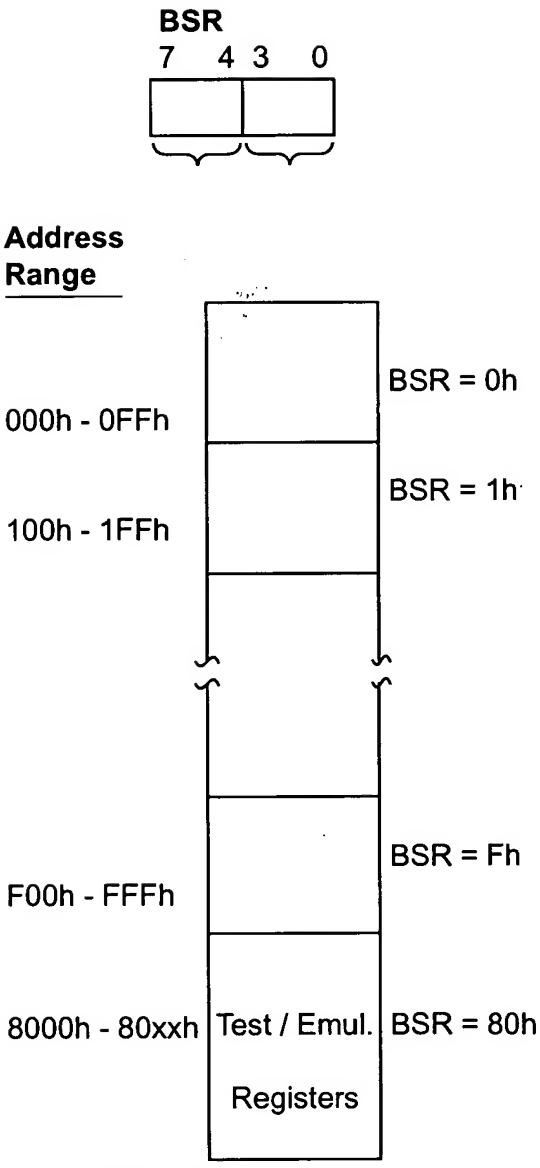
Direct Short Addressing Mode

Figure 46



BSR Operation

Figure 47



BSR Operation During Emulation/Test Modes

Figure 48

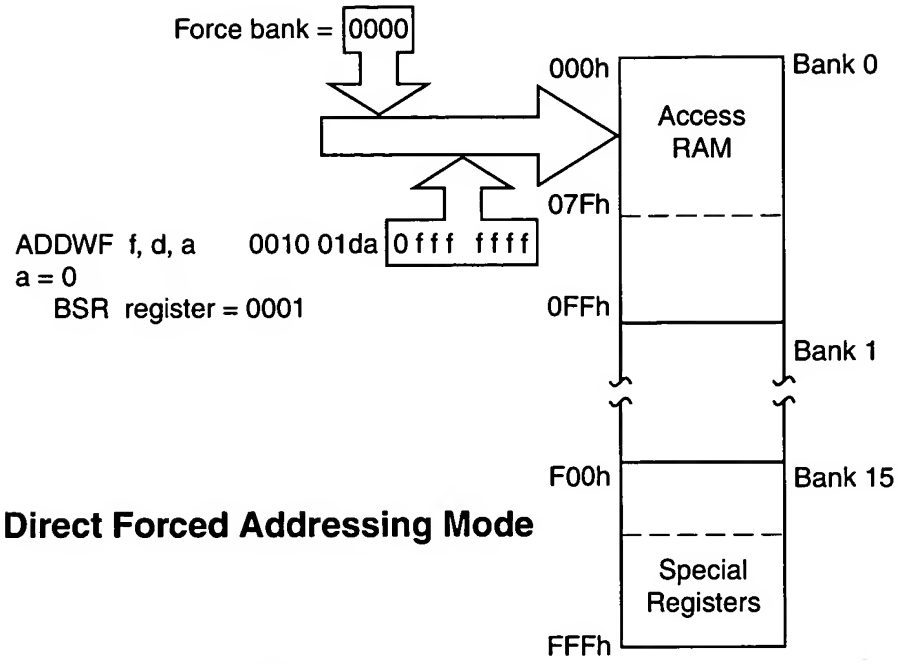


Figure 49

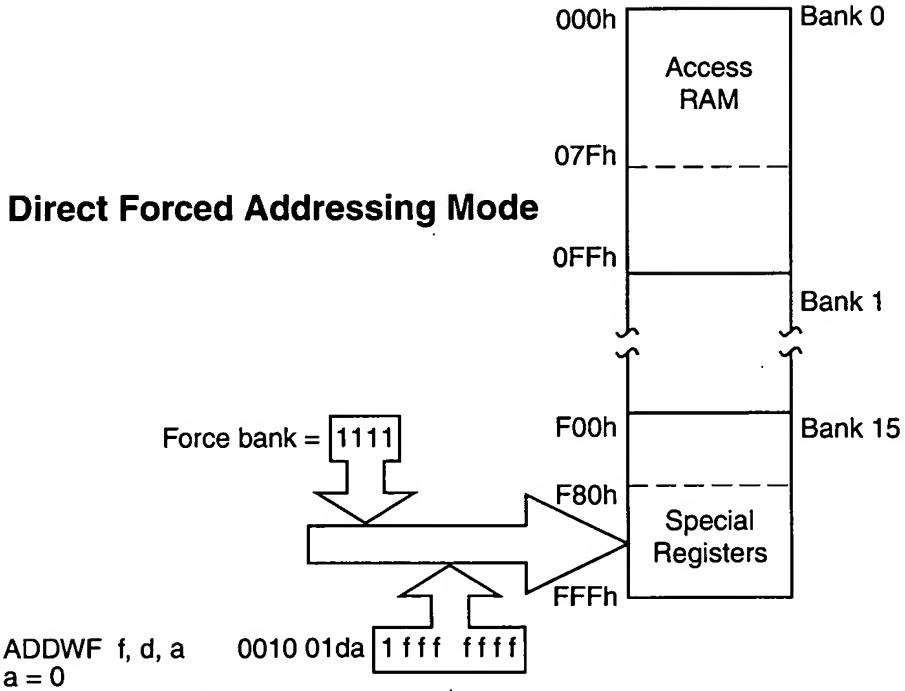
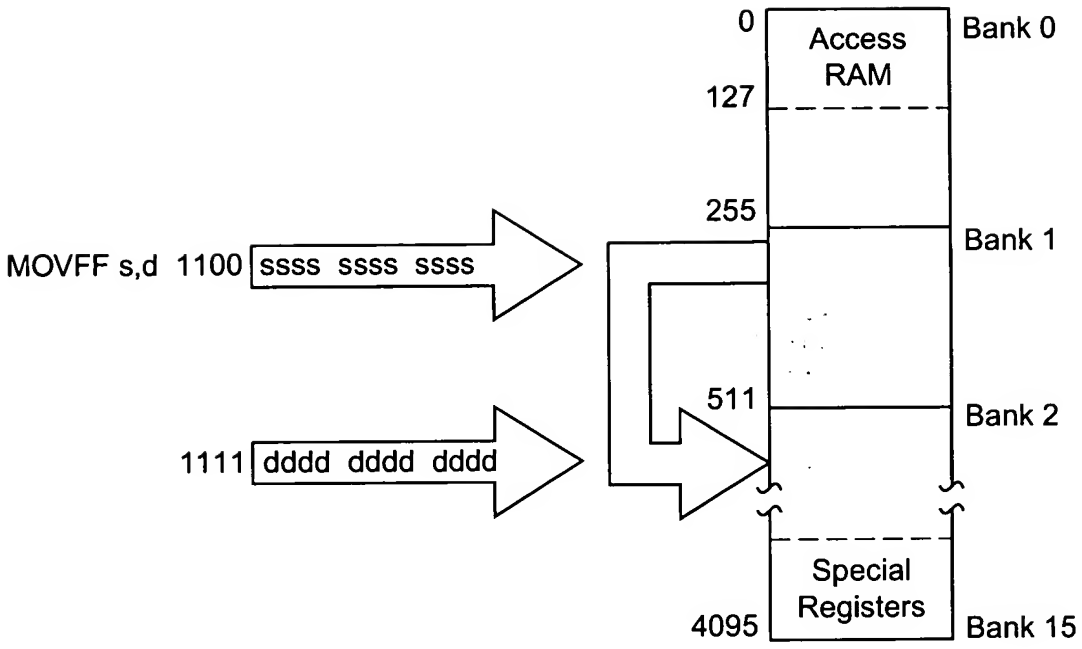
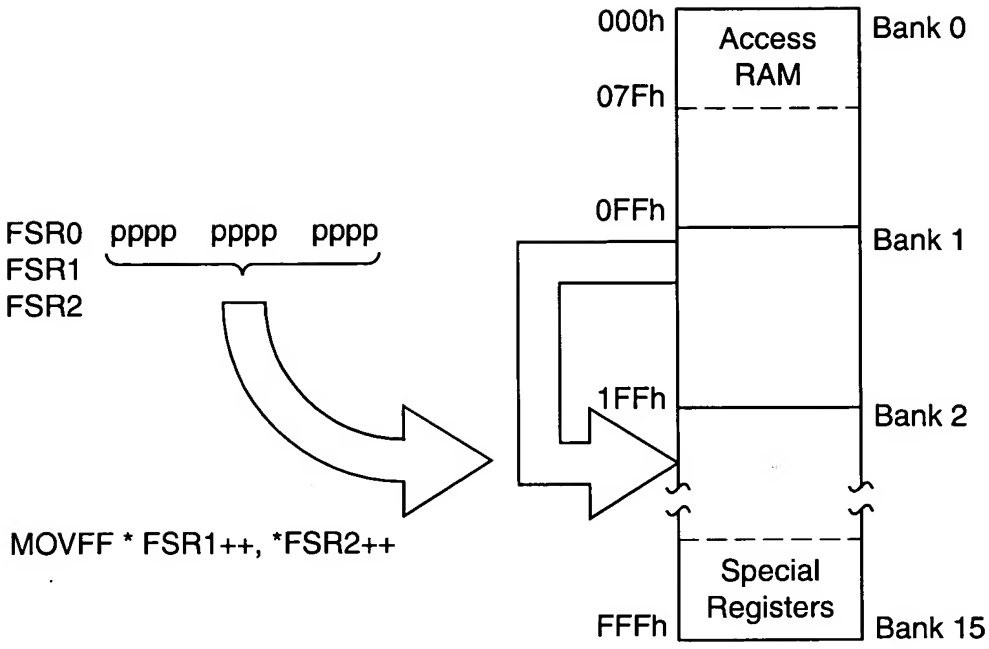


Figure 50



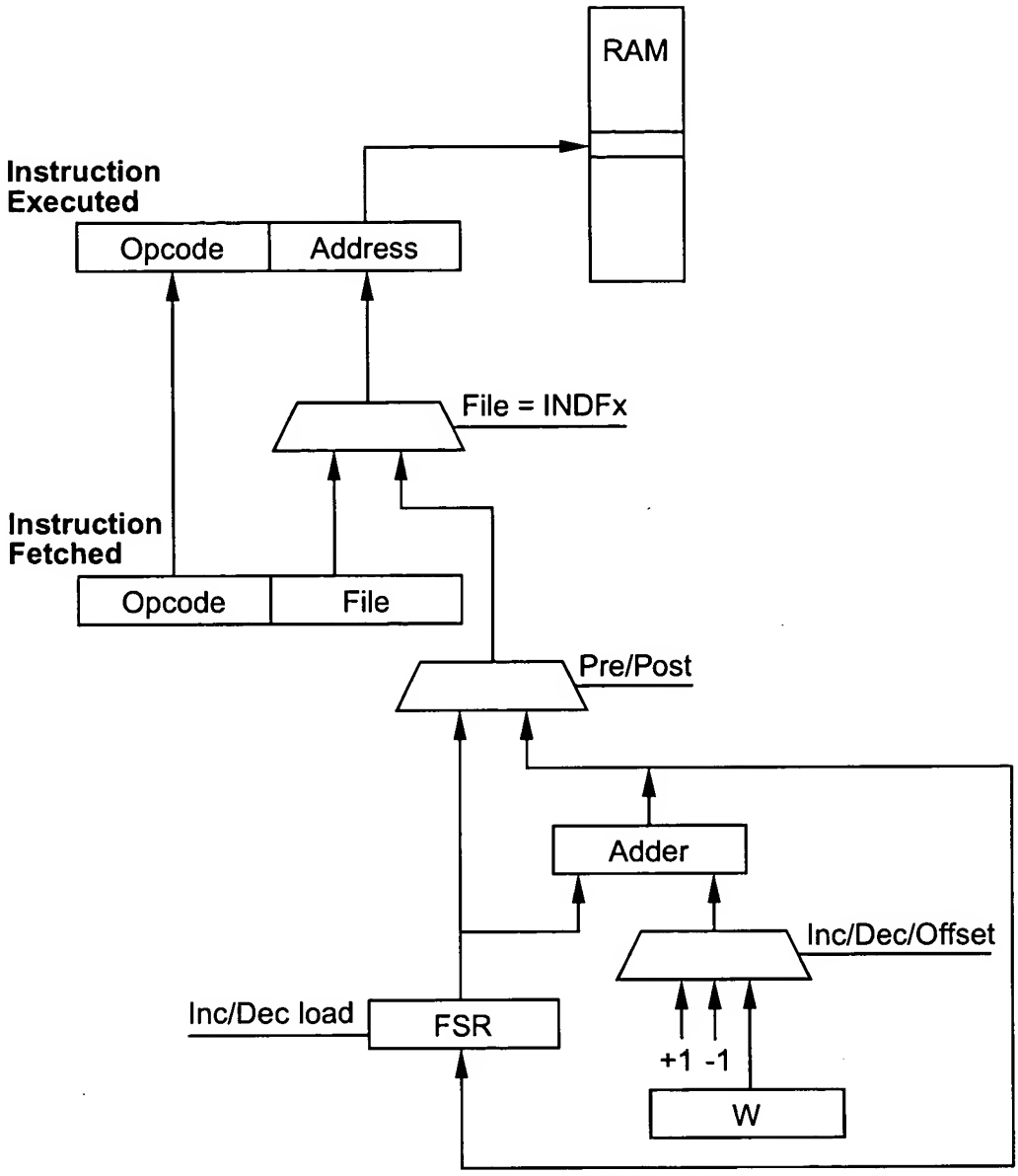
Direct Long Addressing Mode

Figure 51



Indirect Addressing Mode

Figure 52



Indirect Addressing

Figure 53

Field	Description
f f_s f_d	Register file address (00h to FFh) in current or virtual bank, except MOVFF (000h to FFFh)
a	Virtual bank select 0 = override BSR and force virtual bank 1 = do not override BSR Default is a = '1'
s	Fast call/return select 0 = do not update into/from shadow registers 1 = certain registers loaded into/from shadow registers Default is s = '0'
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
n	2's complement number for relative branch instructions
x	Don't care location (= '0' or '1') The assembler will generate code with x = '0'. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select 0 = store result in W 1 = store result in file register f Default is d = '1'
u	Unused, encoded as '0'
label	Label name
C, DC, Z, OV, N	ALU status bits Carry, Digit Carry, Zero, Overflow, Negative
GIE/ GIEH	Global Interrupt Enable bit (INTCON<7>)
PEIE/ GIEL	Low Priority Interrupt Enable bit (INTCON<6>)
TBLPTRU TBLPTRH TBLPTRL	Table Pointer (21-bit)
TABLAT	Table Latch (8-bit)
PRODL	Product of Multiply low byte
PRODH	Product of Multiply high byte
TOSU TOSH TOSL	Top of Stack

Field	Description
PCU PCH PCL	Program Counter
BSR	Bank Select Register
WDT	Watchdog Timer Counter
$\overline{TO}$	Time-out bit
$\overline{PD}$	Power-down bit
dest	Destination either the W register or the specified register file location
*	No Change to TBLPTR
*+	Post-Increment TBLPTR
*-	Post-Decrement TBLPTR
+*	Pre-Increment TBLPTR
[ ]	Options
( )	Contents
→	Assigned to
< >	Register bit field

Opcode Field Descriptions

Figure 54

Field	Description
*FSRn	Selects INDFn Register
*FSRn++	Selects POSTINCn Register
*FSRn--	Selects POSTDECn Register
*(++FSRn)	Selects PREINCn Register
*(FSRn+W)	Selects PLUSWn Register

Indirect Addressing Symbols

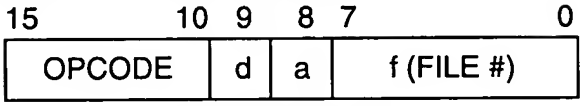
Figure 55

Figures 54 and 55 list the symbols recognized by the MPASM assembler.

Note 1: Any unused opcode is Reserved. Use of any reserved opcode may cause unexpected operation.

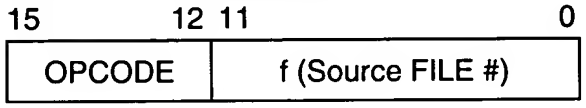
All instruction examples use the following format to represent a hexadecimal number:
0xnn
where 0x signifies a hexadecimal digit.
To represent a binary number:
nnnnnnnnb
where b signifies a binary string.

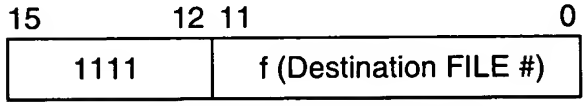
Byte-oriented file register operations



d = 0 for destination W
d = 1 for destination f
a = 0 for force Virtual bank
a = 1 for BSR to select bank
f = 8-bit file register address

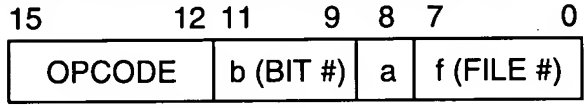
Byte to Byte move operations (2-word)





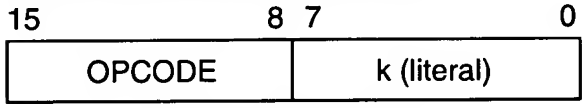
f = 12-bit file register address

Bit-oriented file register operations



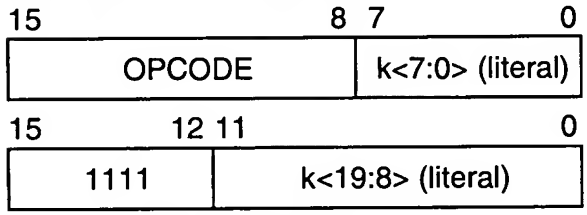
b = 3-bit address
a = 0 for force Virtual bank
a = 1 for BSR to select bank
f = 8-bit file register address

Literal and control operations



k = 8-bit immediate value

CALL and GOTO operations



k = 20-bit immediate value

General Format for Instructions

Figure 56

Mnemonic, Operands	Description	Cycles	16-bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS					
ADDWF f,d,a	ADD W to f	1	0010 01da ffff ffff	C,DC,N,OV,Z	3
ADDWFC f,d,a	ADD W and Carry bit to f	1	0010 00da ffff ffff	C,D,C,N,OV,Z	3
ANDWF f,d,a	AND W with f	1	0001 01da ffff ffff	N,Z	3
CLRF f,a	Clear f	1	0110 101a ffff ffff	Z	3
COMF f,d,a	Complement f	1	0001 11da ffff ffff	N,Z	3
CPFSEQ f,a	Compare f with W, skip if f = W	1 (2)	0110 001a ffff ffff	None	3,5,7,8
CPFSGT f,a	Compare f with W, skip if f > W	1 (2)	0110 010a ffff ffff	None	2,3,5,7,8
CPFSLT f,a	Compare f with W, skip if f< W	1 (2)	0110 000a ffff ffff	None	2,3,5,7,8
DECF f,d,a	Decrement f	1	0000 01da ffff ffff	C,DC,N,OV,Z	3
DECFSZ f,d,a	Decrement f, skip if 0	1 (2)	0010 11da ffff ffff	None	3,5,7,8
DCFSNZ f,d,a	Decrement f, skip if not 0	1 (2)	0100 11da ffff ffff	None	3,5,7,8
INCF f,d,a	Increment f	1	0010 10da ffff ffff	C,DC,N,OV,Z	3
INCFSZ f,d,a	Increment f, skip if 0	1 (2)	0011 11da ffff ffff	None	3,5,7,8
INFSNZ f,d,a	Increment f, skip if not 0	1 (2)	0100 10da ffff ffff	None	3,5,7,8
IORWF f,d,a	Inclusive OR W with f	1	0001 00da ffff ffff	N,Z	3
MOVF f,d,a	Move f	1	0101 00da ffff ffff	N,Z	3
MOVFF f_S,f_d	Move f_S (1st word) to f_d (2nd word)	2	1100 ffff ffff ffff 1111 ffff ffff ffff	None	6
MOVWF f,a	Move W to f	1	0110 111a ffff ffff	None	3
MULWF f,a	Multiply W with f	1	0000 001a ffff ffff	None	3
NEGF f,a	Negate f	1	0110 110a ffff ffff	C,DC,N,OV,Z	1,3
NOP ---	No Operation	1	0000 0000 0000 0000	None	
NOP ---	No Operation (2nd Word)	1	1111 xxxx xxxx xxxx	None	
RLCF f,d,a	Rotate left f though Carry	1	0011 01da ffff ffff	C,N,Z	3
RLNCF f,d,a	Rotate left f (no carry)	1	0100 01da ffff ffff	N,Z	3
RRCF f,d,a	Rotate right f through Carry	1	0011 00da ffff ffff	C,N,Z	3
RRNCF f,d,a	Rotate right f (no carry)	1	0100 00da ffff ffff	N,Z	3
SETF f,a	Set f	1	0110 100a ffff ffff	None	3

Legend: Refer to Table 3-6 for opcode field descriptions.

Note 1: 2's Complement method.

2: Unsigned arithmetic.

3: If a = '0', the Bank Select Register (BSR) will be overridden and Virtual bank is selected: If a = '1', the BSR is used.

4: Multiple cycle instruction for EPROM programming when table pointer selects internal EPROM. The instruction is terminated by an interrupt event. Writing to external program memory is a two-cycle instruction.

5: Two-cycle instruction when condition is true, else single cycle instruction.

6: Two-cycle instruction except for MOVFF to PCL (program counter low byte) in which case it takes 3 cycles.

7: A "skip" means that the instruction fetched during execution of the current instruction is not executed, instead an NOP is executed.

8: When a "skip" instruction executes a skip and is followed by a 2-word instruction, 3 cycles will be executed.

9: If s = '1', certain registers will be loaded from/into shadow registers. If s = '0' no update occurs.

Instruction Set Summary

Figure 57

Mnemonic, Operands		Description	Cycles	16-bit Opcode MSb LSb	Status Affected	Notes
SUBFWB	f,d,a	Subtract f from W with Borrow	1	0101 01da ffff ffff	C,DC,N,OV,Z	1,3
SUBWF	f,d,a	Subtract W from f	1	0101 11da ffff ffff	C,DC,N,OV,Z	1,3
SUBWFB	f,d,a	Subtract W from f with Borrow	1	0101 10da ffff ffff	C,DC,N,OV,Z	1,3
SWAPF	f,d,a	Swap f	1	0011 10da ffff ffff	None	3
TSTFSZ	f,a	Test f, skip if 0	1 (2)	0110 011a ffff ffff	None	3,5,7,8
XORWF	f,d,a	Exclusive OR W with f	1	0001 10da ffff ffff	N,Z	3
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f,b,a	Bit Clear f	1	1001 bbba ffff ffff	None	3
BSF	f,b,a	Bit Set f	1	1000 bbba ffff ffff	None	3
BTFSC	f,b,a	Bit test f, skip if clear	1 (2)	1011 bbba ffff ffff	None	3,5,7,8
BTFSS	f,b,a	Bit test f, skip if set	1 (2)	1010 bbba ffff ffff	None	3,5,7,8
BTG	f,b,a	Bit Toggle f	1	0111 bbba ffff ffff	None	3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	ADD literal to W	1	0000 1111 kkkk kkkk	C,DC,N,OV,Z	
ANDLW	k	AND literal with W	1	0000 1011 kkkk kkkk	N,Z	
BC	n	Branch if Carry	1 (2)	1110 0010 nnnn nnnn	None	
BN	n	Branch if Negative	1 (2)	1110 0110 nnnn nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110 0011 nnnn nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110 0111 nnnn nnnn	None	
BNV	n	Branch if Not Overflow	1 (2)	1110 0101 nnnn nnnn	None	
BNZ	n	Branch if Not Zero	1 (2)	1110 0001 nnnn nnnn	None	
BRA	n	Unconditional branch	2	1101 0nnn nnnn nnnn	None	
BV	n	Branch if Overflow	1 (2)	1110 0100 nnnn nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110 0000 nnnn nnnn	None	
CALL	k,s	Subroutine Call (1st word) (2nd word)	2	1110 110s kkkk kkkk 1111 kkkk kkkk kkkk	None	9
CLRWDT	-	Clear Watchdog Timer	1	0000 0000 0000 0100	$\overline{TO}$,$\overline{PD}$	
DAW	-	Decimal Adjust W Register	1	0000 0000 0000 0111	C	
GOTO	k	Unconditional Branch (1st word) (2nd word)	2	1110 1111 kkkk kkkk 1111 kkkk kkkk kkkk	None	
HALT	-	Halt processor	1	0000 0000 0000 0001	None	
IORLW	k	Inclusive OR literal with W	1	0000 1001 kkkk kkkk	N,Z	

Legend: Refer to Table 3-6 for opcode field descriptions.

Note 1: 2's Complement method.

2: Unsigned arithmetic.

3: If a = '0', the Bank Select Register (BSR) will be overridden and Virtual bank is selected: If a = '1', the BSR is used.

4: Multiple cycle instruction for EPROM programming when table pointer selects internal EPROM. The instruction is terminated by an interrupt event. Writing to external program memory is a two-cycle instruction.

5: Two-cycle instruction when condition is true, else single cycle instruction.

6: Two-cycle instruction except for MOVFF to PCL (program counter low byte) in which case it takes 3 cycles.

7: A "skip" means that the instruction fetched during execution of the current instruction is not executed, instead an NOP is executed.

8: When a "skip" instruction executes a skip and is followed by a 2-word instruction, 3 cycles will be executed.

9: If s = '1', certain registers will be loaded from/into shadow registers. If s = '0' no update occurs.

Instruction Set Summary (Continued)

Figure 58

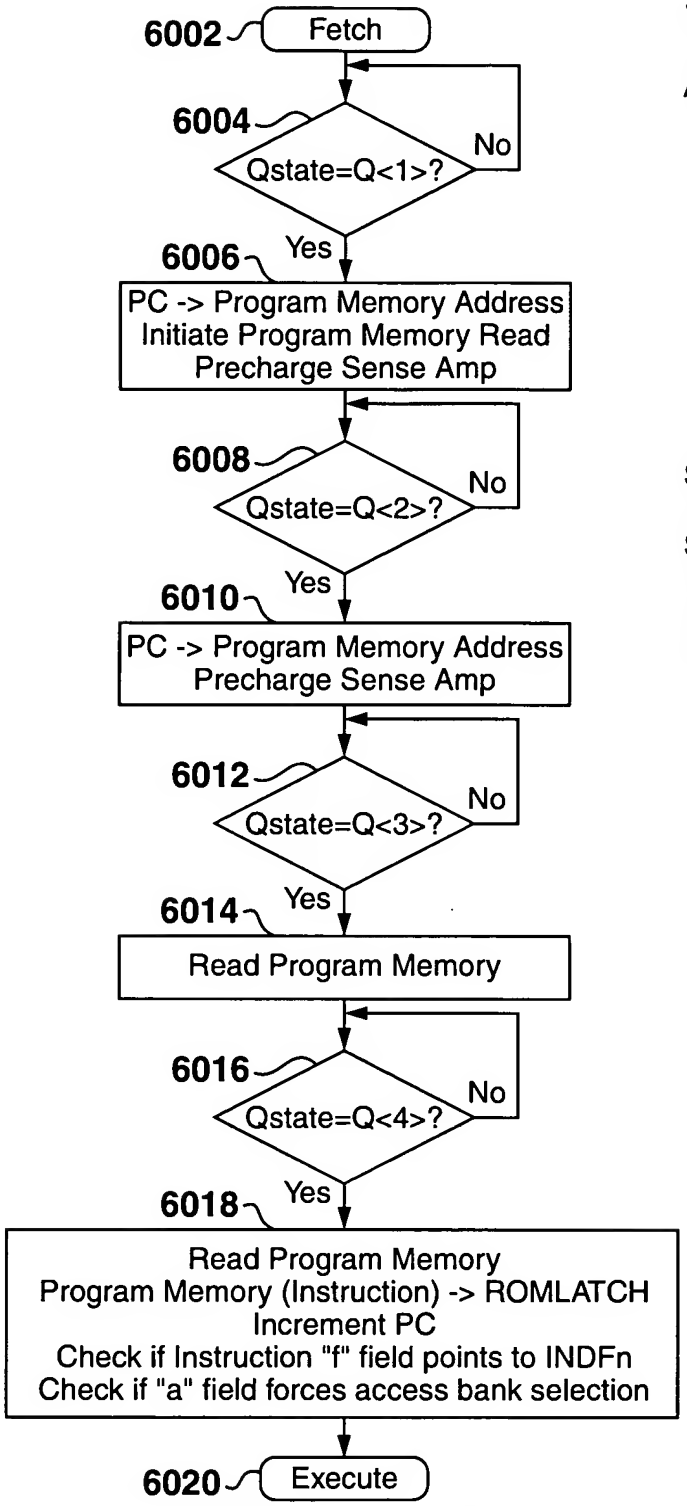
Mnemonic, Operands	Description	Cycles	16-bit Opcode MSb LSb	Status Affected	Notes
LFSR f,k	Move Literal to FSR (second word)	2	1110 1110 00ff kkkk 1111 0000 kkkk kkkk	None	
MOVLB k	Move literal to low nibble in BSR	1	0000 0001 0000 kkkk	None	
MOVLW k	Move literal to W	1	0000 1110 kkkk kkkk	None	
MULLW k	Multiply literal with W	1	0000 1101 kkkk kkkk	None	
POP ---	Pop Top of return stack (TOS)	1	0000 0000 0000 0110	None	
PUSH ---	Push Top of return stack (TOS)	1	0000 0000 0000 0101	None	
RCALL n	Unconditional subroutine branch	2	1101 1nnn nnnn nnnn	None	
RESET ---	Generate a Reset (same as MCLR reset)	1	0000 0000 1111 1111	All - Reset	
RETFIE s	Return from interrupt (and enable interupts)	2	0000 0000 0001 000s	GIEH,GIEL All if s=1	9
RETLW k	Return literal to W	2	0000 1100 kkkk kkkk	None	
RETURN s	Return from Subroutine	2	0000 0000 0001 001s	None if s=0 All if s=1	9
SLEEP ---	Enter SLEEP Mode	1	0000 0000 0000 0011	$\overline{TO}$, $\overline{PD}$	
SUBLW k	Subtract W from literal	1	0000 1000 kkkk kkkk	N,OV,C,DC,Z	
TBLRD* ---	Table Read (no change to TBLPTR)	2	0000 0000 0000 1000	None	
TBLRD*+ ---	Table Read (post-increment TBLPTR)	2	0000 0000 0000 1001	None	
TBLRD*- ---	Table Read (post-decrement TBLPTR)	2	0000 0000 0000 1010	None	
TBLRD+* ---	Table Read (pre-increment TBLPTR)	2	0000 0000 0000 1011	None	
TBLWT* ---	Table Write (no change to TBLPTR)	2	0000 0000 0000 1100	None	4
TBLWT*+ ---	Table Write (post-increment TBLPTR)	2	0000 0000 0000 1101	None	4
TBLWT*- ---	Table Write (post-decrement TBLPTR)	2	0000 0000 0000 1110	None	4
TBLWT+* ---	Table Write (pre-increment TBLPTR)	2	0000 0000 0000 1111	None	4
XORLW k	Exclusive OR literal with W	1	0000 1010 kkkk kkkk	N,Z	

Legend: Refer to Table 3-6 for opcode field descriptions.

Note 1: 2's Complement method.
2: Unsigned arithmetic.
3: If a = '0', the Bank Select Register (BSR) will be overridden and Virtual bank is selected: If a = '1', the BSR is used.
4: Multiple cycle instruction for EPROM programming when table pointer selects internal EPROM. The instruction is terminated by an interrupt event. Writing to external program memory is a two-cycle instruction.
5: Two-cycle instruction when condition is true, else single cycle instruction.
6: Two-cycle instruction except for MOVFF to PCL (program counter low byte) in which case it takes 3 cycles.
7: A "skip" means that the instruction fetched during execution of the current instruction is not executed, instead an NOP is executed.
8: When a "skip" instruction executes a skip and is followed by a 2-word instruction, 3 cycles will be executed.
9: If s = '1', certain registers will be loaded from/into shadow registers. If s = '0' no update occurs.

Instruction Set Summary (Continued)

Figure 59



Valid For:
ADDWF
ADDWFC
ANDWF
COMF
DECF
INCF
IORWF
MOVF
RLCF
RLNCF
RRCF
RRNCF
SUBFWB
SUBWF
SUBWFB
SWAPF
XORWF
MOVWF
NOP

Figure 60

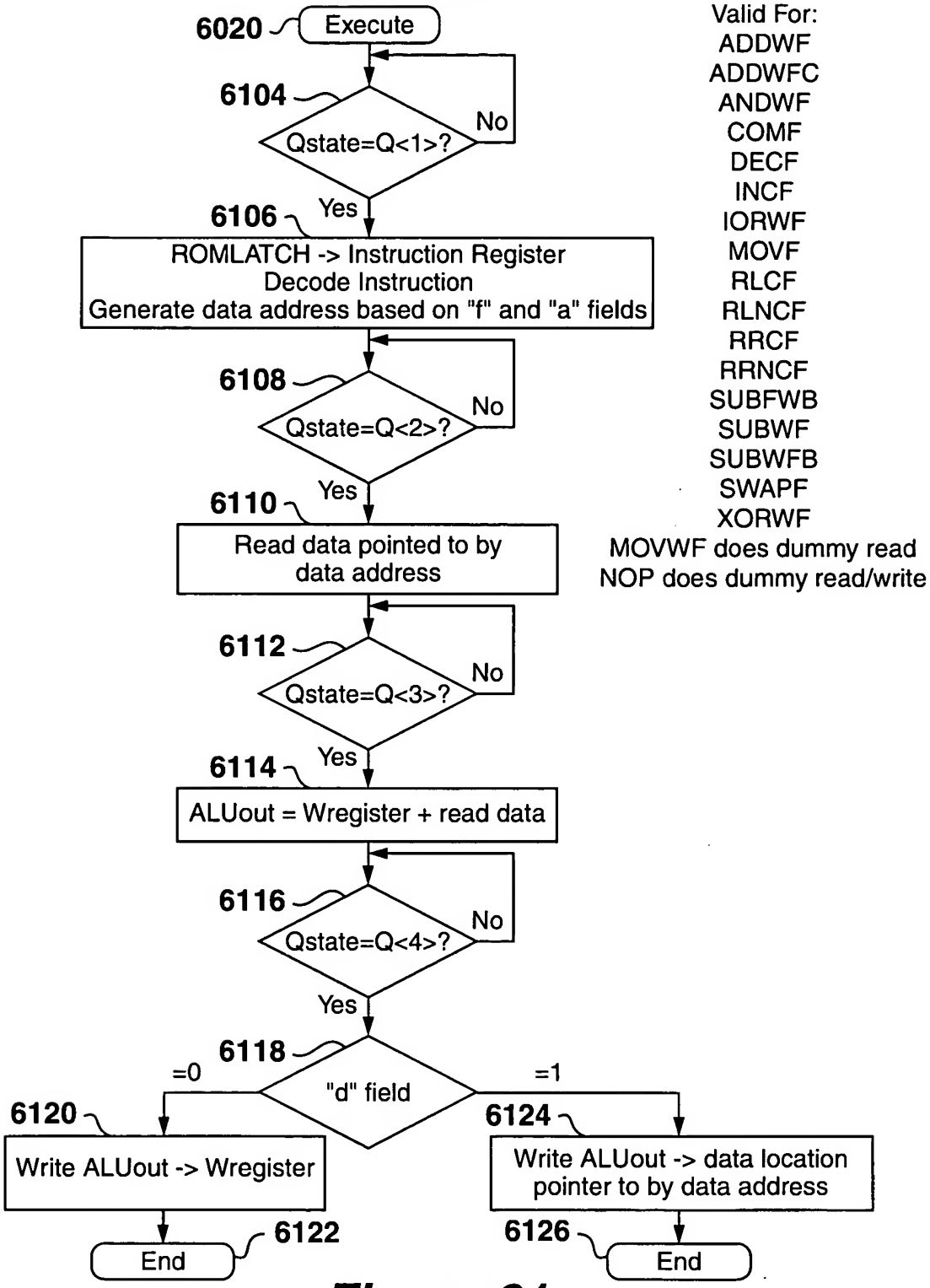

6020
Execute
6104
Qstate=Q<1>?
No
Yes
6106
ROMLATCH -> Instruction Register
Decode Instruction
Generate data address based on "f" and "a" fields
6108
Qstate=Q<2>?
No
Yes
6110
Read data pointed to by
data address
6112
Qstate=Q<3>?
No
Yes
6114
ALUout = Wregister + read data
6116
Qstate=Q<4>?
No
Yes
6118
"d" field
=0
=1
6120
Write ALUout -> Wregister
6124
Write ALUout -> data location
pointer to by data address
6122
End
6126
End
Valid For:
ADDWF
ADDWFC
ANDWF
COMF
DECF
INCF
IORWF
MOVF
RLCF
RLNCF
RRCF
RRNCF
SUBFWB
SUBWF
SUBWFB
SWAPF
XORWF
MOVWF does dummy read
NOP does dummy read/write

Figure 61

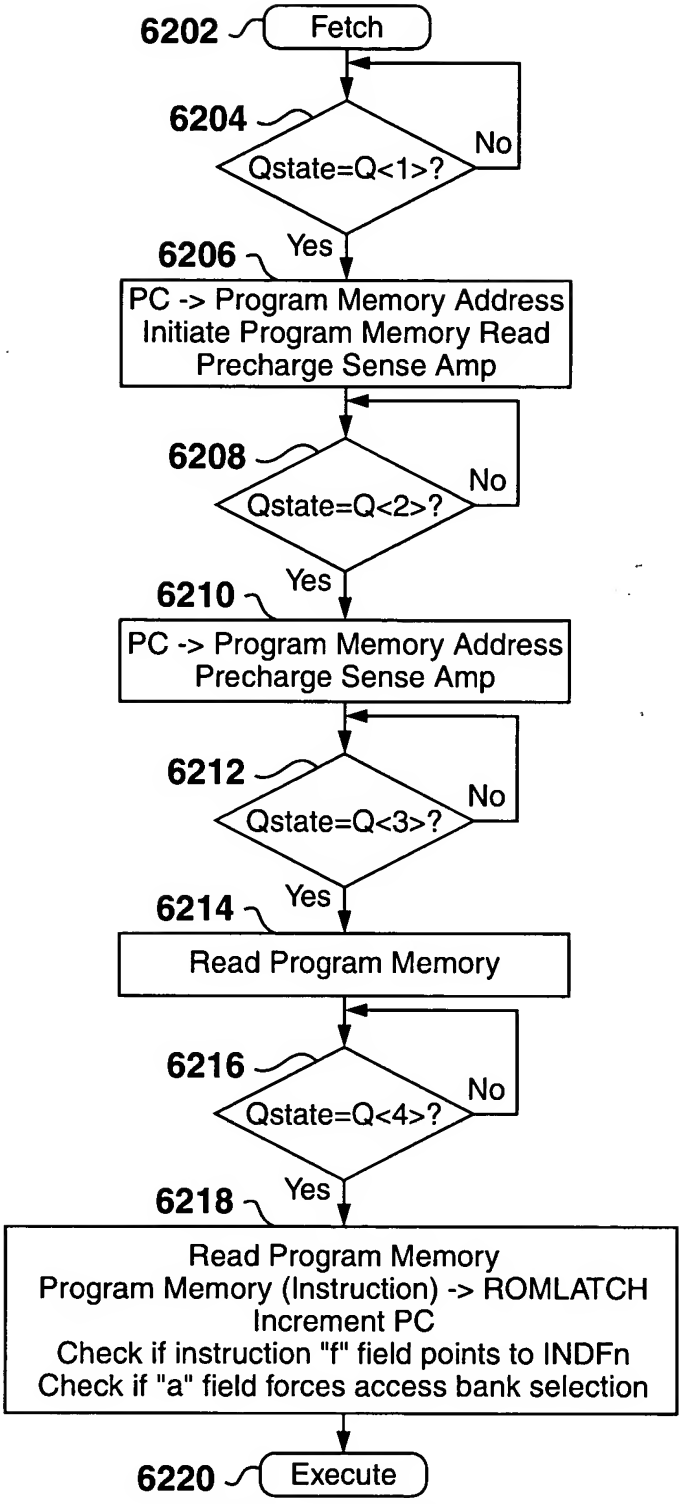

6202
Fetch
6204
Qstate=Q<1>?
No
Yes
6206
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
6208
Qstate=Q<2>?
No
Yes
6210
PC -> Program Memory Address
Precharge Sense Amp
6212
Qstate=Q<3>?
No
Yes
6214
Read Program Memory
6216
Qstate=Q<4>?
No
Yes
6218
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
Check if instruction "f" field points to INDFn
Check if "a" field forces access bank selection
6220
Execute

Figure 62

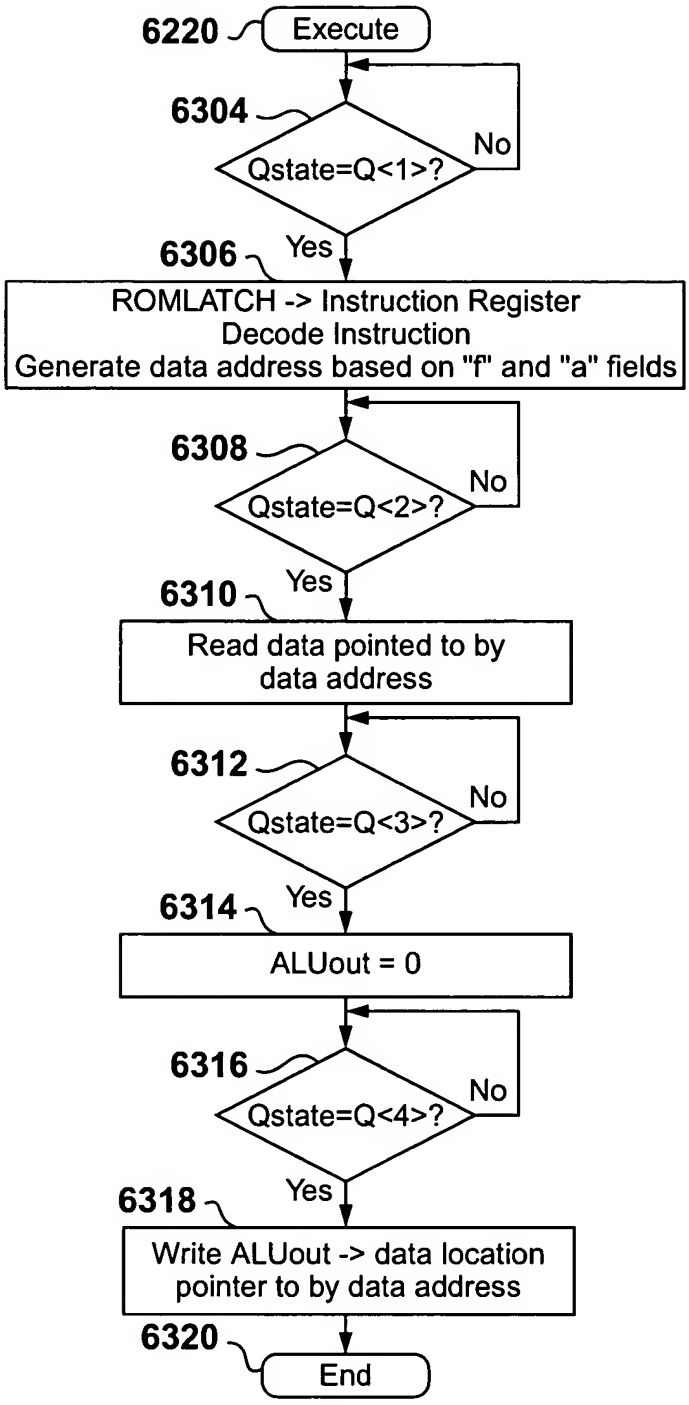

6220
Execute
6304
Qstate=Q<1>?
No
Yes
6306
ROMLATCH -> Instruction Register
Decode Instruction
Generate data address based on "f" and "a" fields
6308
Qstate=Q<2>?
No
Yes
6310
Read data pointed to by
data address
6312
Qstate=Q<3>?
No
Yes
6314
ALUout = 0
6316
Qstate=Q<4>?
No
Yes
6318
Write ALUout -> data location
pointer to by data address
6320
End

Figure 63

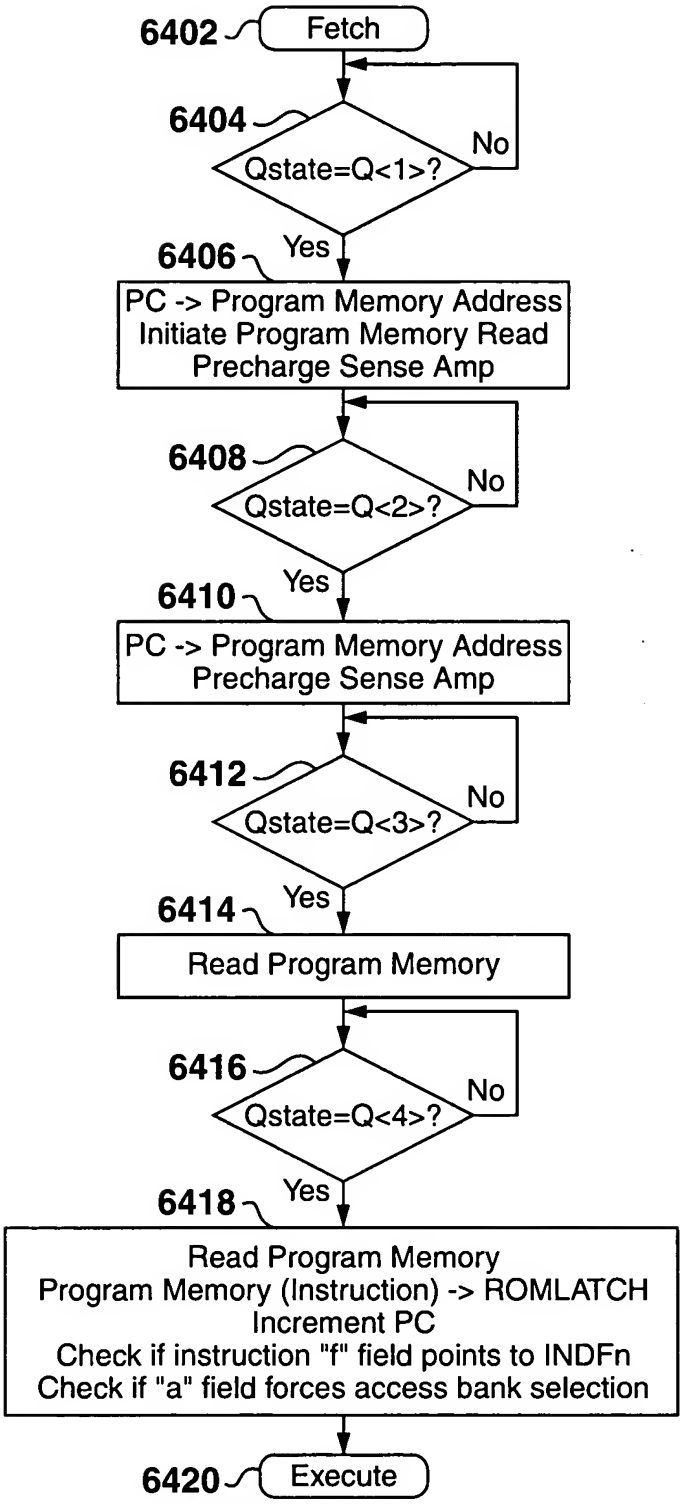

6402
Fetch
6404
Qstate=Q<1>?
No
Yes
6406
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
6408
Qstate=Q<2>?
No
Yes
6410
PC -> Program Memory Address
Precharge Sense Amp
6412
Qstate=Q<3>?
No
Yes
6414
Read Program Memory
6416
Qstate=Q<4>?
No
Yes
6418
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
Check if instruction "f" field points to INDFn
Check if "a" field forces access bank selection
6420
Execute

Figure 64

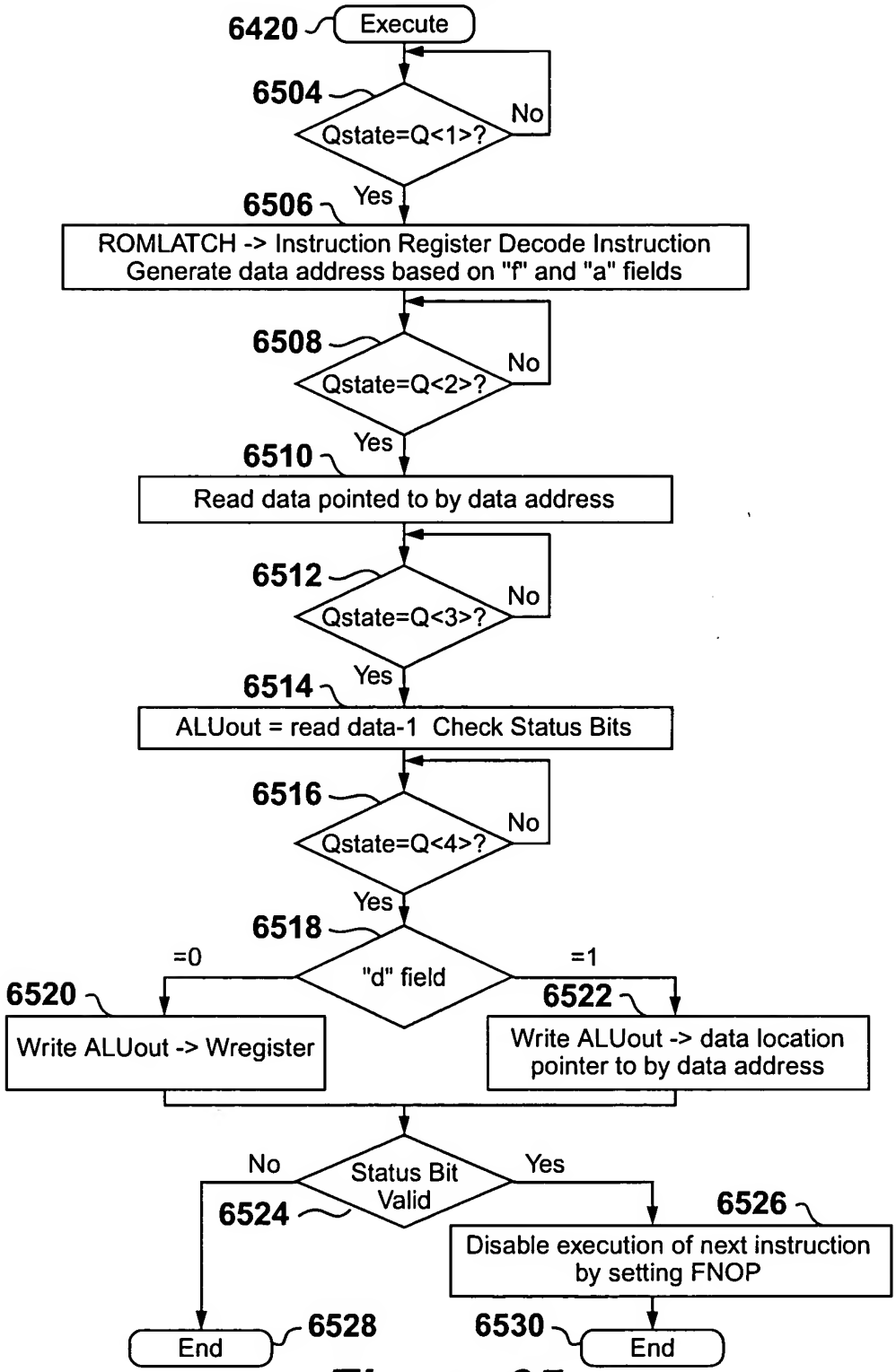

6420
Execute
6504
Qstate=Q<1>?
No
Yes
6506
ROMLATCH -> Instruction Register Decode Instruction
Generate data address based on "f" and "a" fields
6508
Qstate=Q<2>?
No
Yes
6510
Read data pointed to by data address
6512
Qstate=Q<3>?
No
Yes
6514
ALUout = read data-1 Check Status Bits
6516
Qstate=Q<4>?
No
Yes
6518
"d" field
=0
=1
6520
Write ALUout -> Wregister
6522
Write ALUout -> data location
pointer to by data address
Status Bit
Valid
No
Yes
6524
6526
Disable execution of next instruction
by setting FNOP
End
6528
6530
End

Figure 65

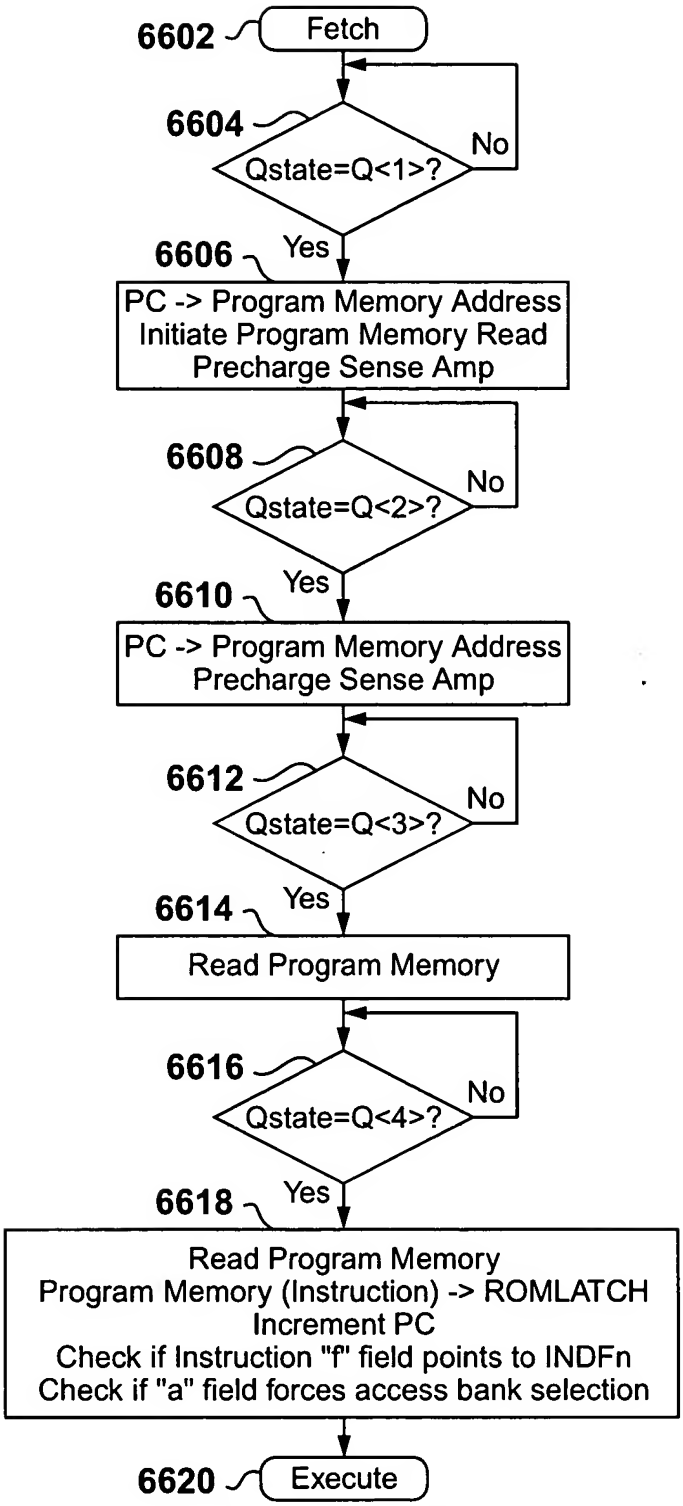

6602
Fetch
6604
Qstate=Q<1>?
No
Yes
6606
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
6608
Qstate=Q<2>?
No
Yes
6610
PC -> Program Memory Address
Precharge Sense Amp
6612
Qstate=Q<3>?
No
Yes
6614
Read Program Memory
6616
Qstate=Q<4>?
No
Yes
6618
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
Check if Instruction "f" field points to INDFn
Check if "a" field forces access bank selection
6620
Execute

Figure 66

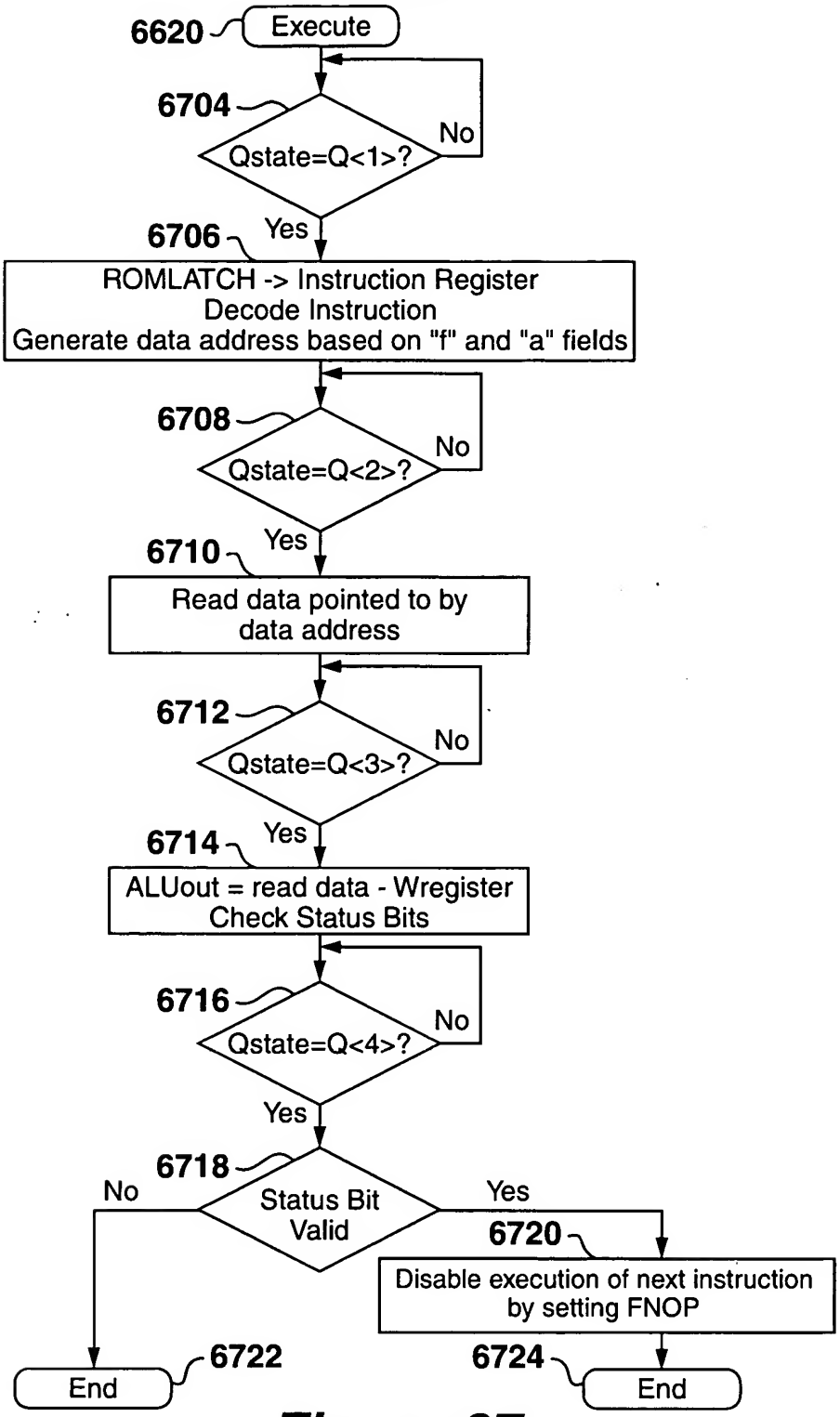

6620
Execute
6704
Qstate=Q<1>?
No
Yes
6706
ROMLATCH -> Instruction Register
Decode Instruction
Generate data address based on "f" and "a" fields
6708
Qstate=Q<2>?
No
Yes
6710
Read data pointed to by
data address
6712
Qstate=Q<3>?
No
Yes
6714
ALUout = read data - Wregister
Check Status Bits
6716
Qstate=Q<4>?
No
Yes
6718
Status Bit
Valid
No
Yes
6720
Disable execution of next instruction
by setting FNOP
6722
End
6724
End

Figure 67

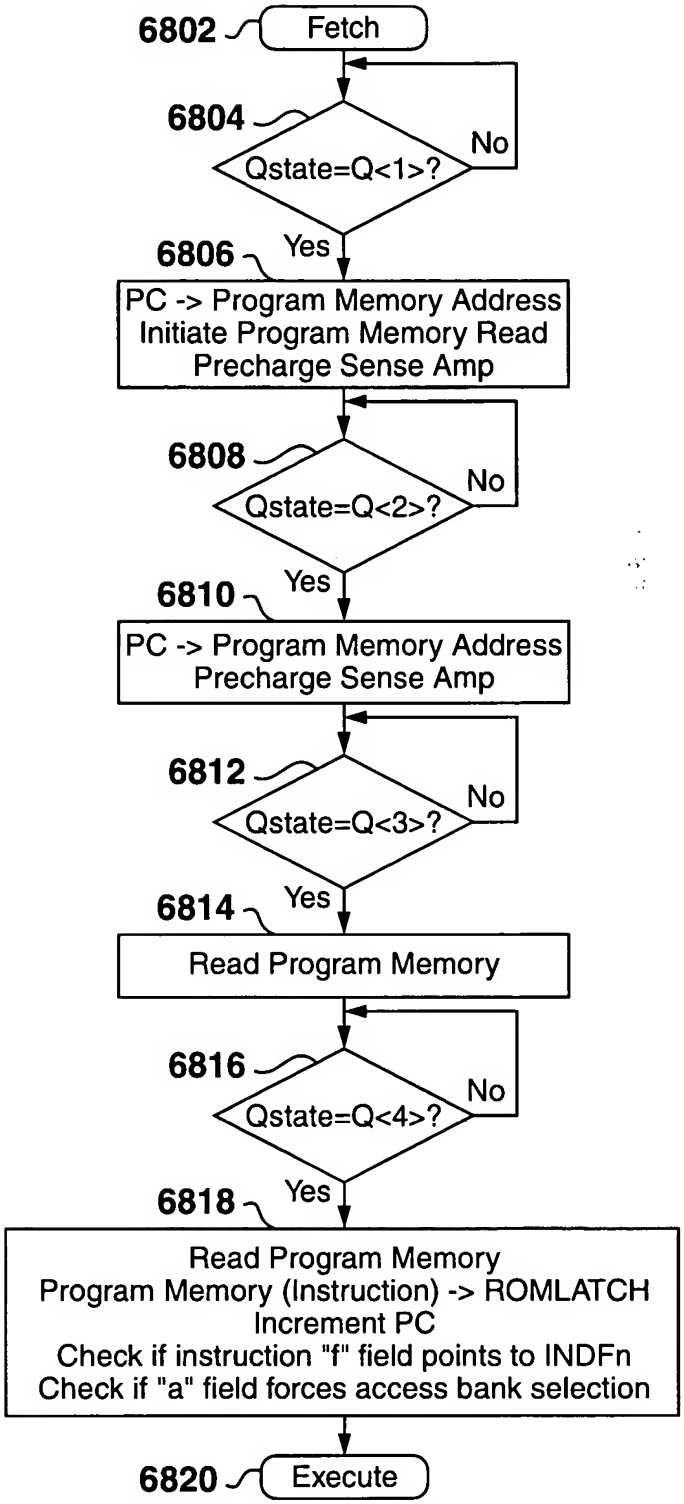

6802
Fetch
6804
Qstate=Q<1>?
No
Yes
6806
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
6808
Qstate=Q<2>?
No
Yes
6810
PC -> Program Memory Address
Precharge Sense Amp
6812
Qstate=Q<3>?
No
Yes
6814
Read Program Memory
6816
Qstate=Q<4>?
No
Yes
6818
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
Check if instruction "f" field points to INDFn
Check if "a" field forces access bank selection
6820
Execute

Figure 68

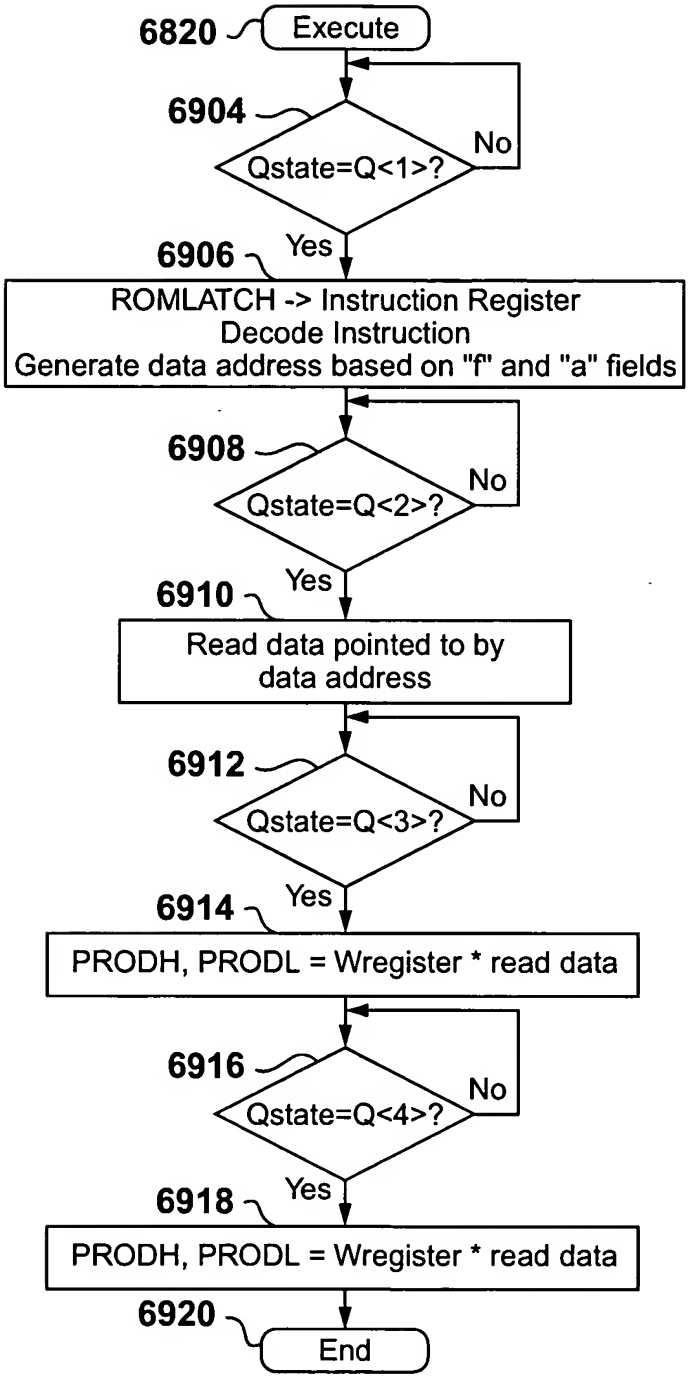

6820
Execute
6904
Qstate=Q<1>?
No
Yes
6906
ROMLATCH -> Instruction Register
Decode Instruction
Generate data address based on "f" and "a" fields
6908
Qstate=Q<2>?
No
Yes
6910
Read data pointed to by
data address
6912
Qstate=Q<3>?
No
Yes
6914
PRODH, PRODL = Wregister * read data
6916
Qstate=Q<4>?
No
Yes
6918
PRODH, PRODL = Wregister * read data
6920
End

Figure 69

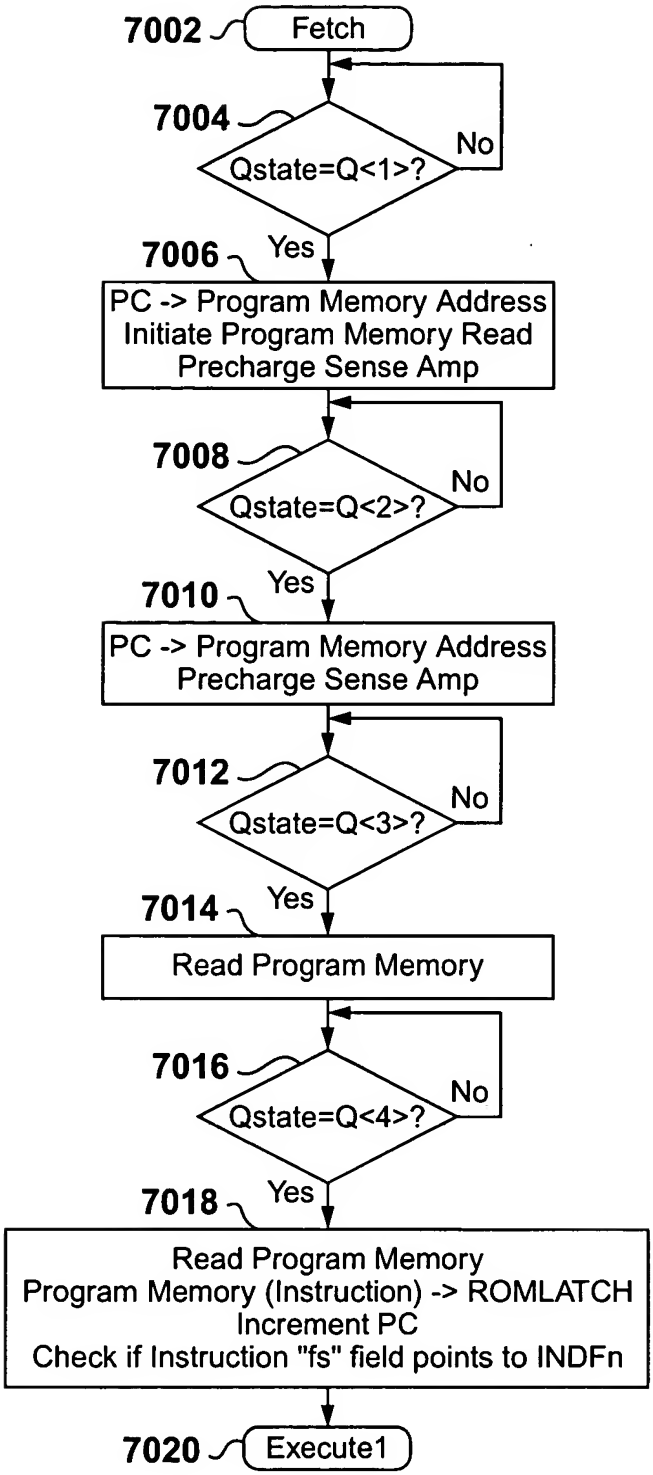

7002
Fetch
7004
Qstate=Q<1>?
No
Yes
7006
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
7008
Qstate=Q<2>?
No
Yes
7010
PC -> Program Memory Address
Precharge Sense Amp
7012
Qstate=Q<3>?
No
Yes
7014
Read Program Memory
7016
Qstate=Q<4>?
No
Yes
7018
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
Check if Instruction "fs" field points to INDFn
7020
Execute1

Figure 70

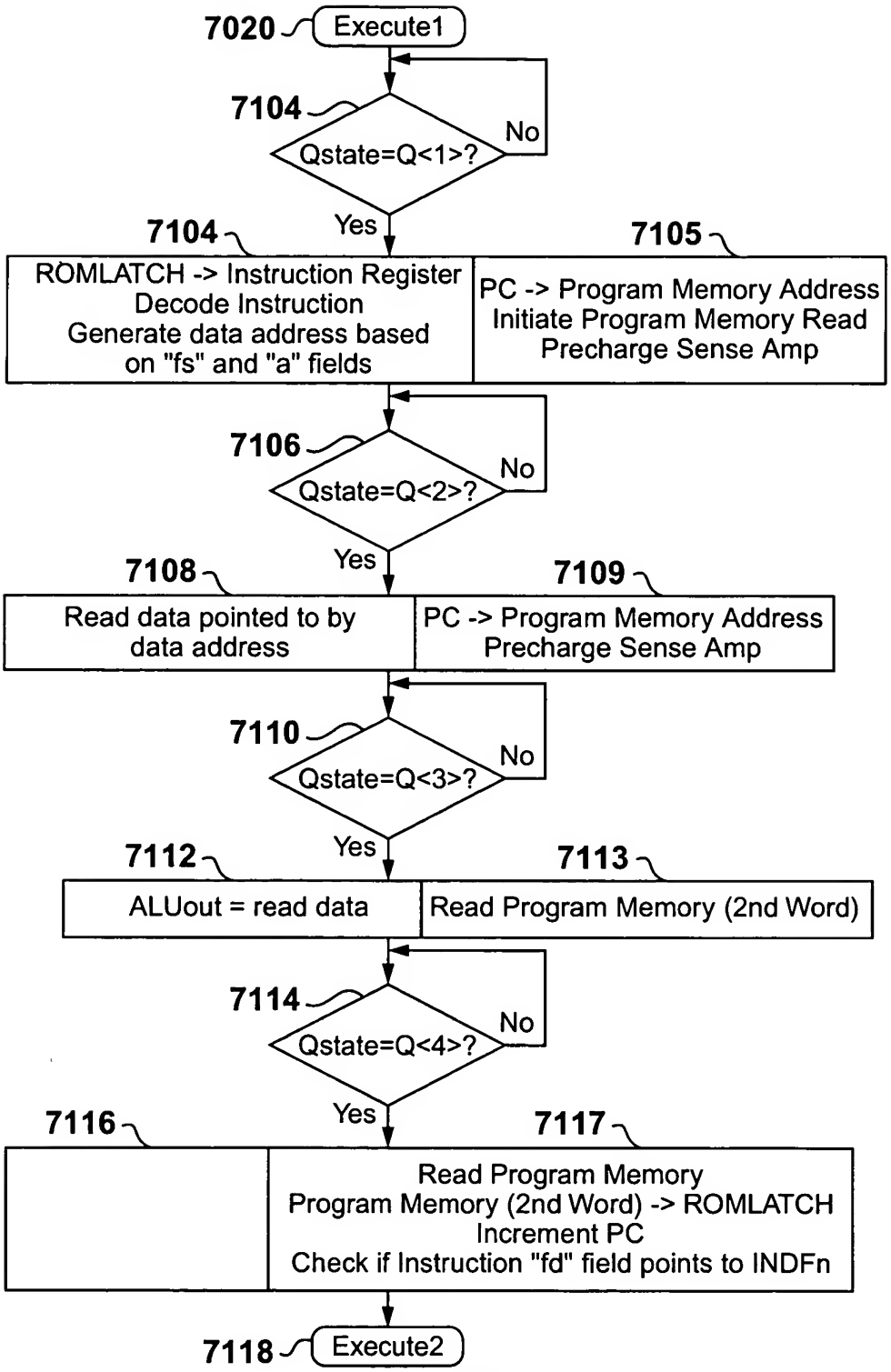

7020
Execute1
7104
Qstate=Q<1>?
No
Yes
7104
ROMLATCH -> Instruction Register
Decode Instruction
Generate data address based on "fs" and "a" fields
7105
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
7106
Qstate=Q<2>?
No
Yes
7108
Read data pointed to by data address
7109
PC -> Program Memory Address
Precharge Sense Amp
7110
Qstate=Q<3>?
No
Yes
7112
ALUout = read data
7113
Read Program Memory (2nd Word)
7114
Qstate=Q<4>?
No
Yes
7116
7117
Read Program Memory
Program Memory (2nd Word) -> ROMLATCH
Increment PC
Check if Instruction "fd" field points to INDFn
7118
Execute2

Figure 71

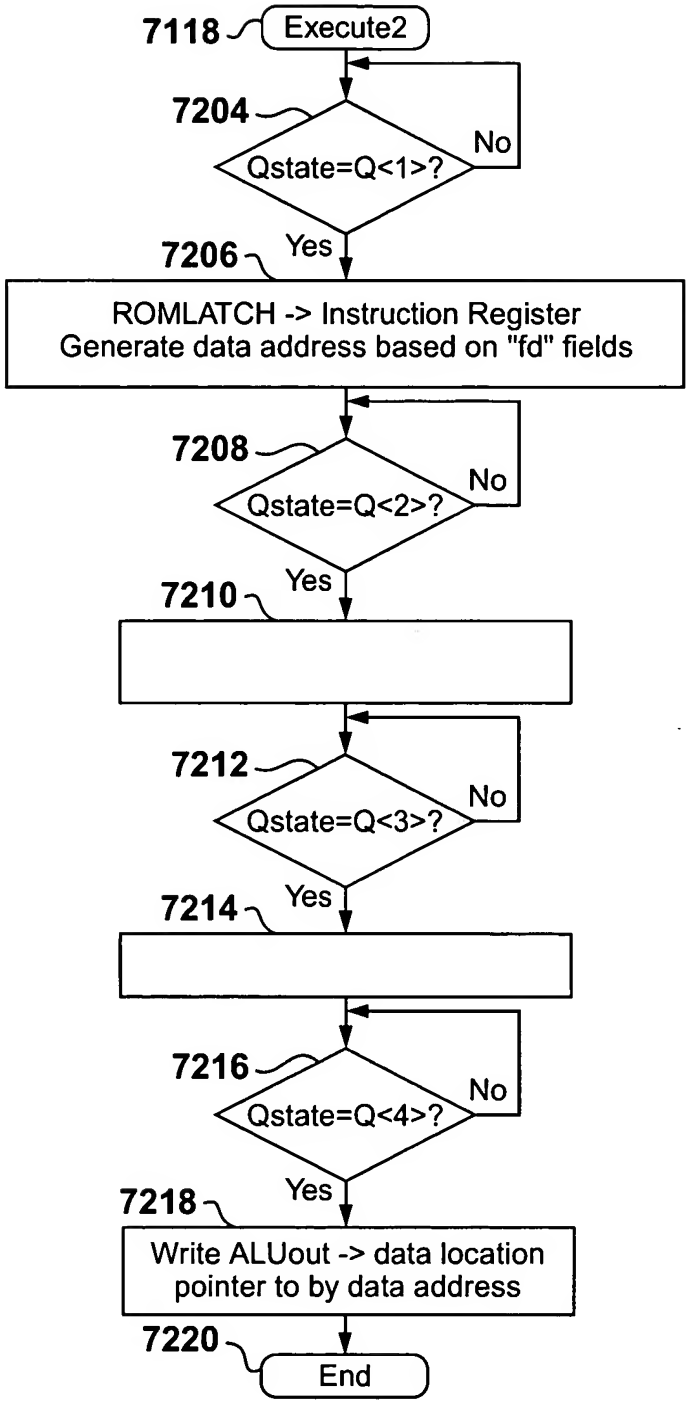

7118
Execute2
7204
Qstate=Q<1>?
No
Yes
7206
ROMLATCH -> Instruction Register
Generate data address based on "fd" fields
7208
Qstate=Q<2>?
No
Yes
7210
7212
Qstate=Q<3>?
No
Yes
7214
7216
Qstate=Q<4>?
No
Yes
7218
Write ALUout -> data location
pointer to by data address
7220
End

Figure 72

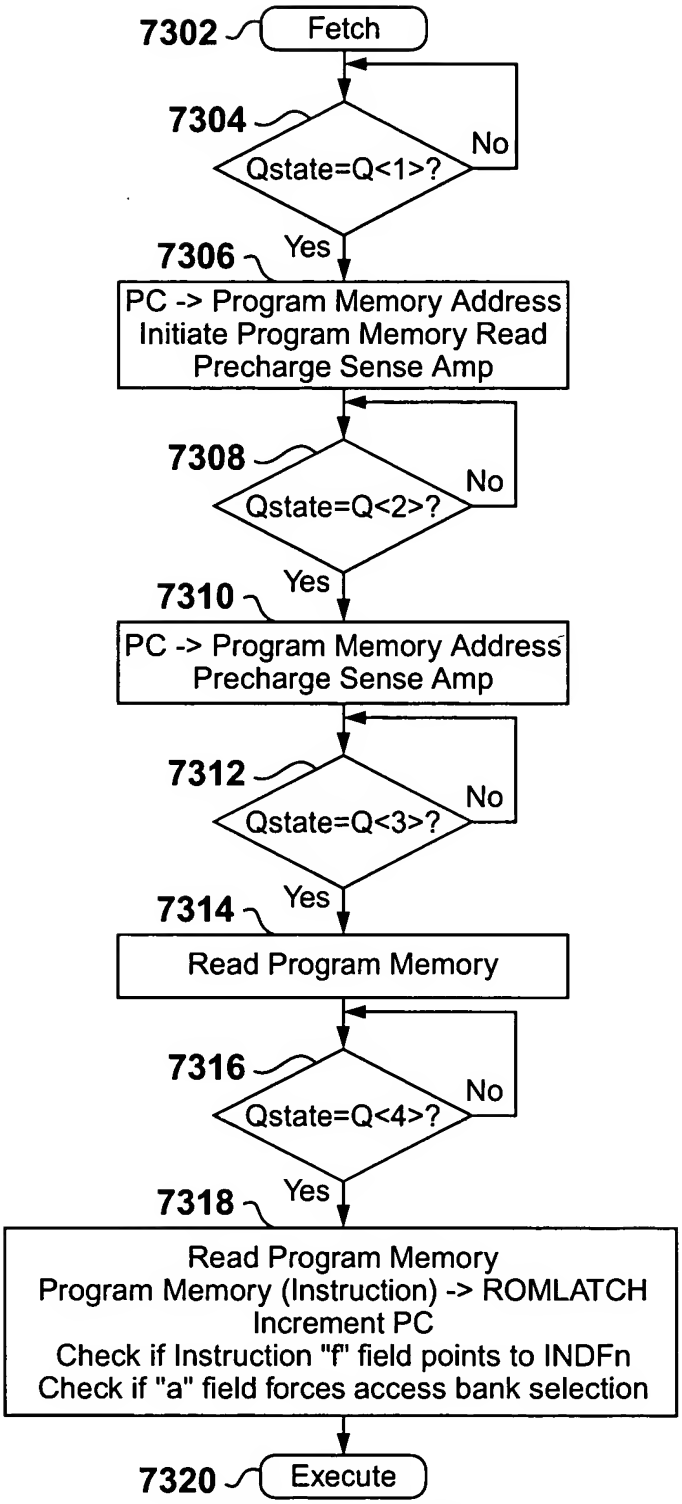

7302
Fetch
7304
Qstate=Q<1>?
No
Yes
7306
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
7308
Qstate=Q<2>?
No
Yes
7310
PC -> Program Memory Address
Precharge Sense Amp
7312
Qstate=Q<3>?
No
Yes
7314
Read Program Memory
7316
Qstate=Q<4>?
No
Yes
7318
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
Check if Instruction "f" field points to INDFn
Check if "a" field forces access bank selection
7320
Execute

Figure 73

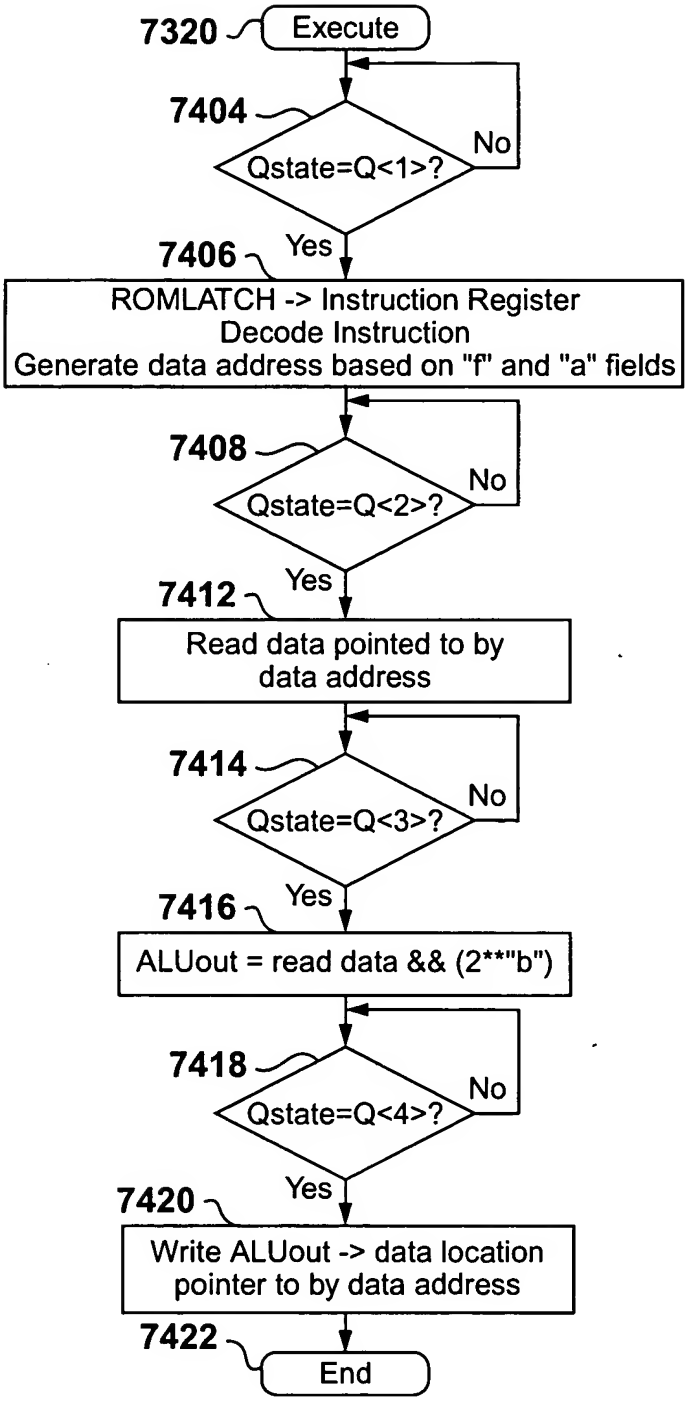

7320
Execute
7404
Qstate=Q<1>?
No
Yes
7406
ROMLATCH -> Instruction Register
Decode Instruction
Generate data address based on "f" and "a" fields
7408
Qstate=Q<2>?
No
Yes
7412
Read data pointed to by
data address
7414
Qstate=Q<3>?
No
Yes
7416
ALUout = read data && (2**"b")
7418
Qstate=Q<4>?
No
Yes
7420
Write ALUout -> data location
pointer to by data address
7422
End

Figure 74

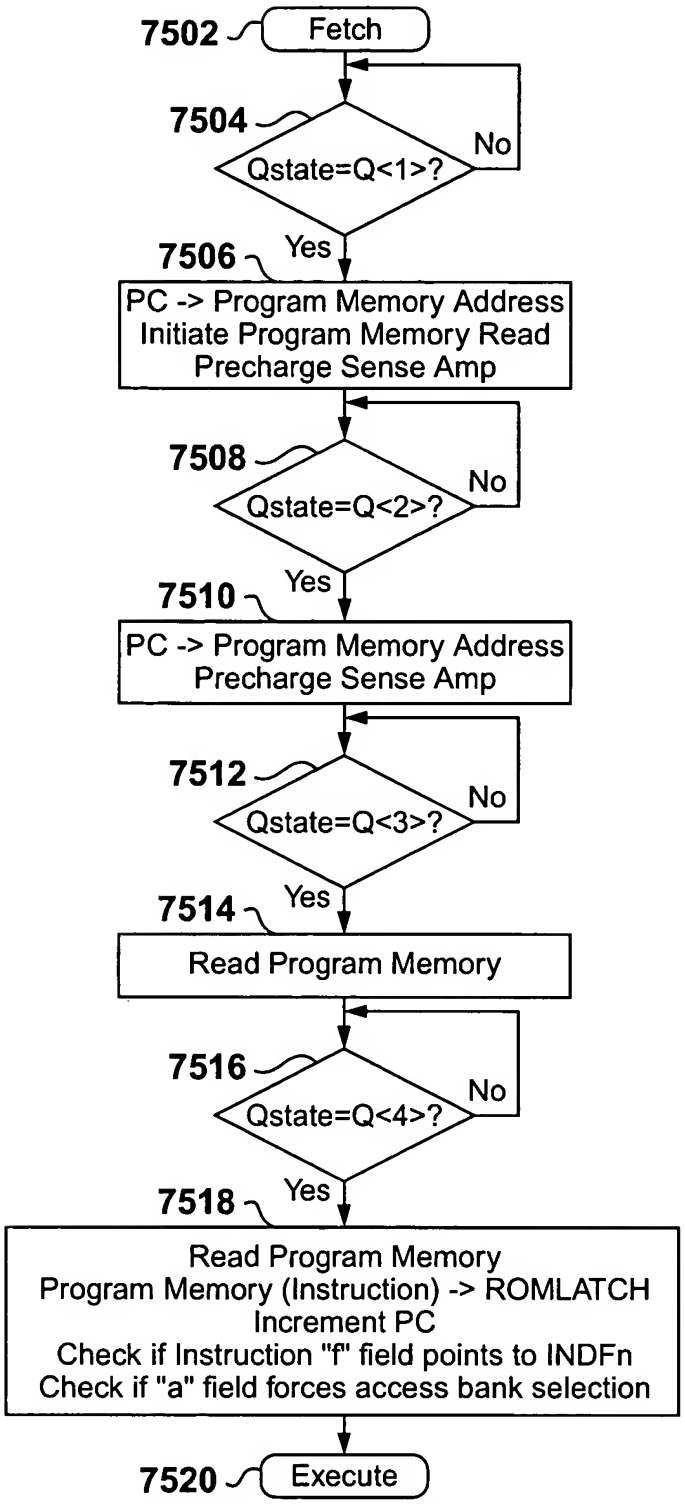

7502
Fetch
7504
Qstate=Q<1>?
No
Yes
7506
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
7508
Qstate=Q<2>?
No
Yes
7510
PC -> Program Memory Address
Precharge Sense Amp
7512
Qstate=Q<3>?
No
Yes
7514
Read Program Memory
7516
Qstate=Q<4>?
No
Yes
7518
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
Check if Instruction "f" field points to INDFn
Check if "a" field forces access bank selection
7520
Execute

Figure 75

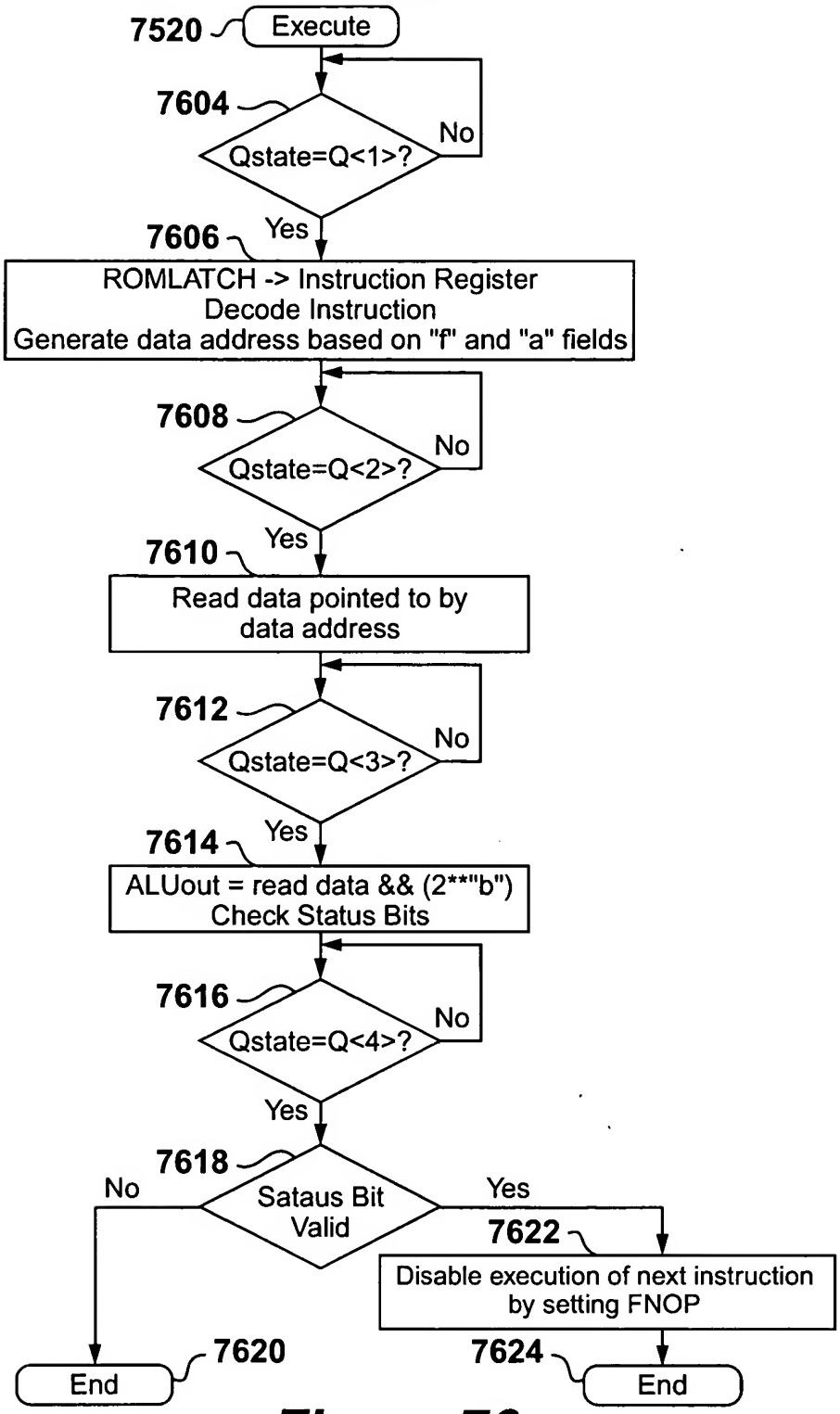

7520
Execute
7604
Qstate=Q<1>?
No
Yes
7606
ROMLATCH -> Instruction Register
Decode Instruction
Generate data address based on "f" and "a" fields
7608
Qstate=Q<2>?
No
Yes
7610
Read data pointed to by
data address
7612
Qstate=Q<3>?
No
Yes
7614
ALUout = read data && (2**"b")
Check Status Bits
7616
Qstate=Q<4>?
No
Yes
7618
Sataus Bit
Valid
No
Yes
7622
Disable execution of next instruction
by setting FNOP
7620
End
7624
End

Figure 76

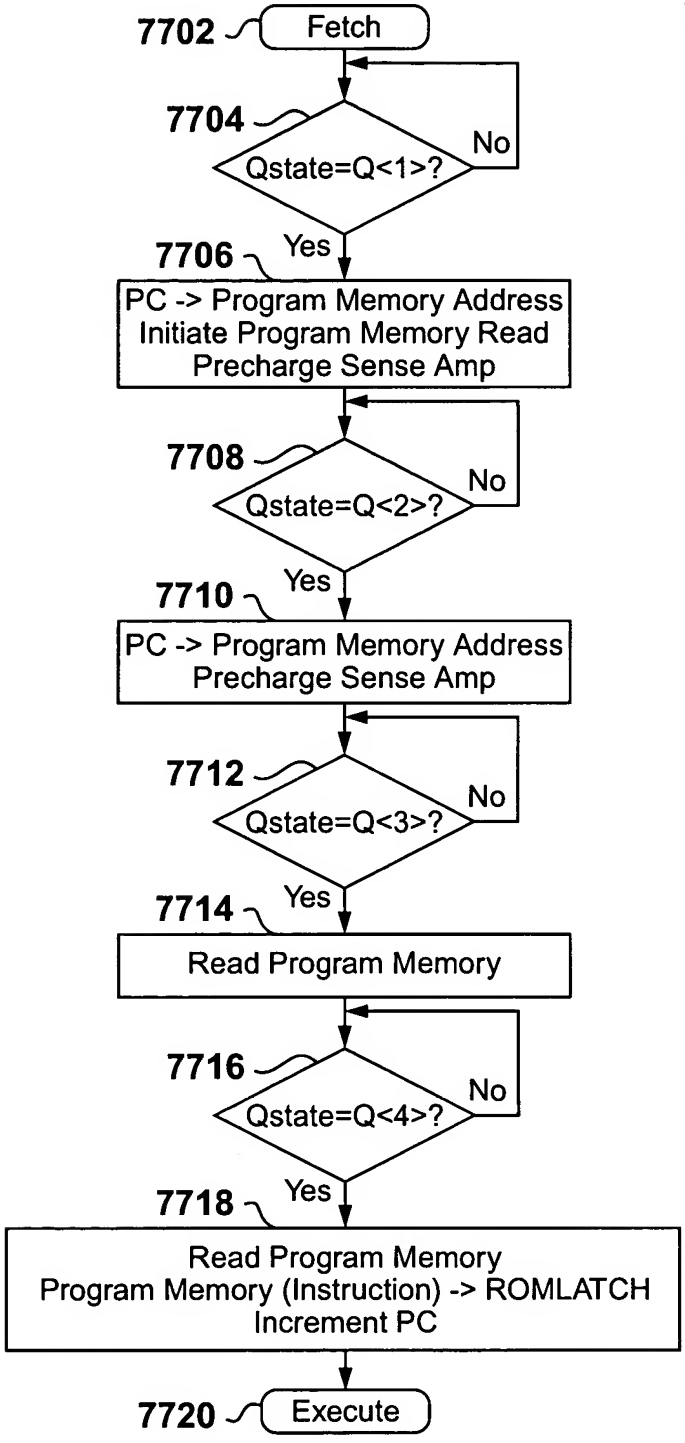

7702
Fetch
7704
Qstate=Q<1>?
No
Yes
7706
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
7708
Qstate=Q<2>?
No
Yes
7710
PC -> Program Memory Address
Precharge Sense Amp
7712
Qstate=Q<3>?
No
Yes
7714
Read Program Memory
7716
Qstate=Q<4>?
No
Yes
7718
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
7720
Execute
Valid For:
ADDLW
ANDLW
IORLW
MOVLW
SUBLW
XORWF

Figure 77

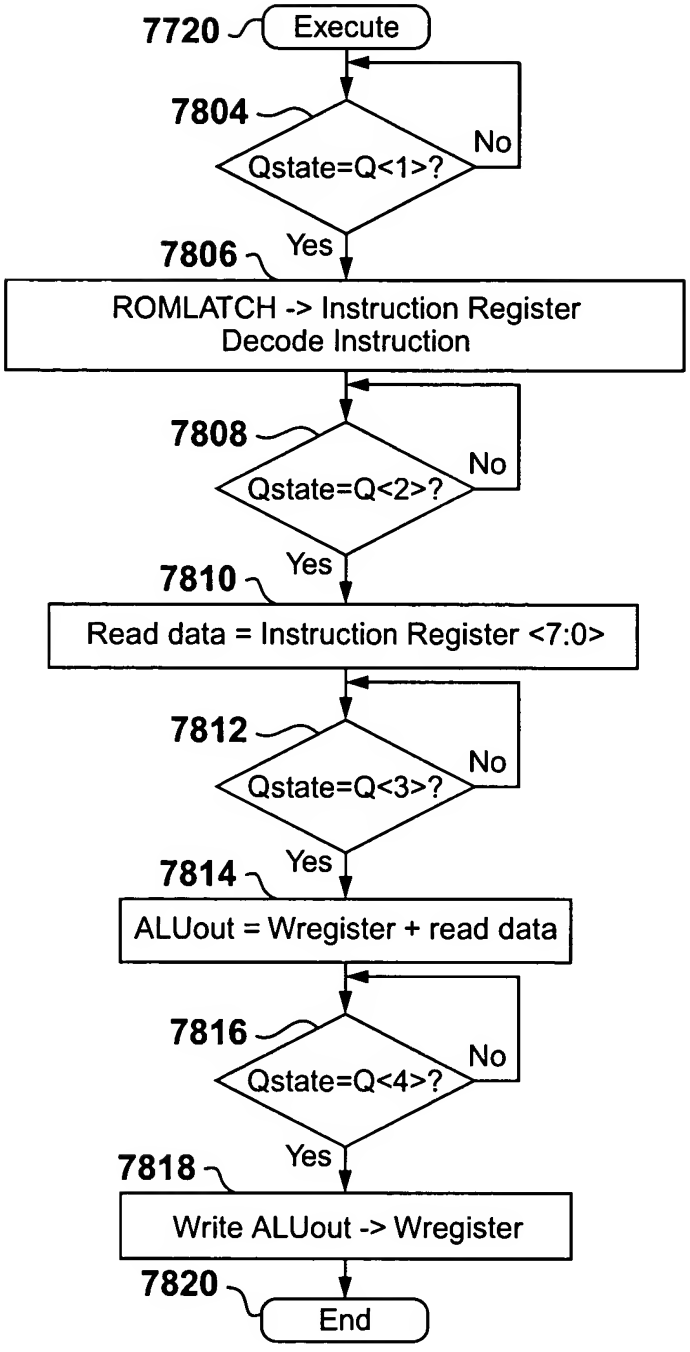

7720
Execute
7804
Qstate=Q<1>?
No
Yes
7806
ROMLATCH -> Instruction Register
Decode Instruction
7808
Qstate=Q<2>?
No
Yes
7810
Read data = Instruction Register <7:0>
7812
Qstate=Q<3>?
No
Yes
7814
ALUout = Wregister + read data
7816
Qstate=Q<4>?
No
Yes
7818
Write ALUout -> Wregister
7820
End
Valid For:
ADDLW
ANDLW
IORLW
MOVLW
SUBLW
XORLW

Figure 78

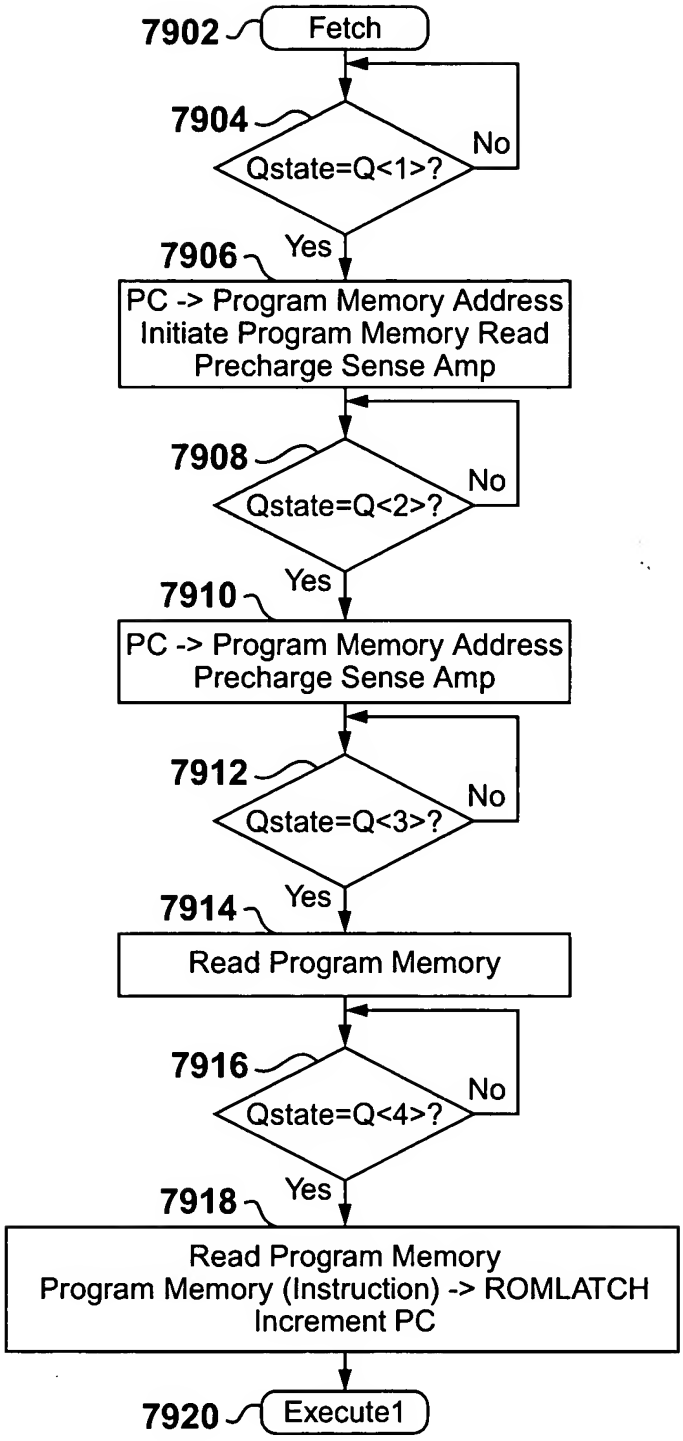

7902
Fetch
7904
Qstate=Q<1>?
No
Yes
7906
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
7908
Qstate=Q<2>?
No
Yes
7910
PC -> Program Memory Address
Precharge Sense Amp
7912
Qstate=Q<3>?
No
Yes
7914
Read Program Memory
7916
Qstate=Q<4>?
No
Yes
7918
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
7920
Execute1

Figure 79

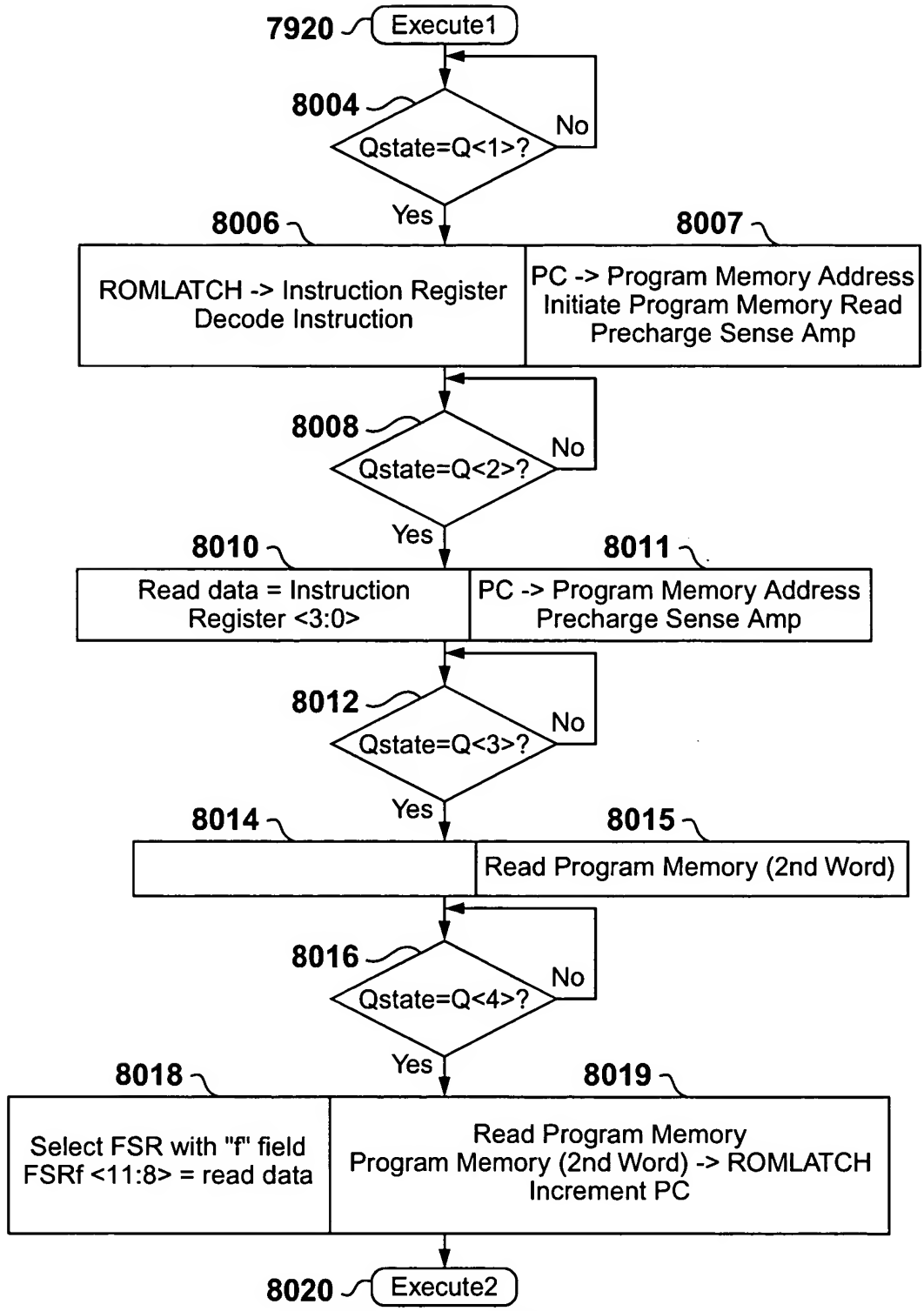

7920
Execute1
8004
Qstate=Q<1>?
No
Yes
8006
ROMLATCH -> Instruction Register
Decode Instruction
8007
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
8008
Qstate=Q<2>?
No
Yes
8010
Read data = Instruction
Register <3:0>
8011
PC -> Program Memory Address
Precharge Sense Amp
8012
Qstate=Q<3>?
No
Yes
8014
8015
Read Program Memory (2nd Word)
8016
Qstate=Q<4>?
No
Yes
8018
Select FSR with "f" field
FSRf <11:8> = read data
8019
Read Program Memory
Program Memory (2nd Word) -> ROMLATCH
Increment PC
8020
Execute2

Figure 80

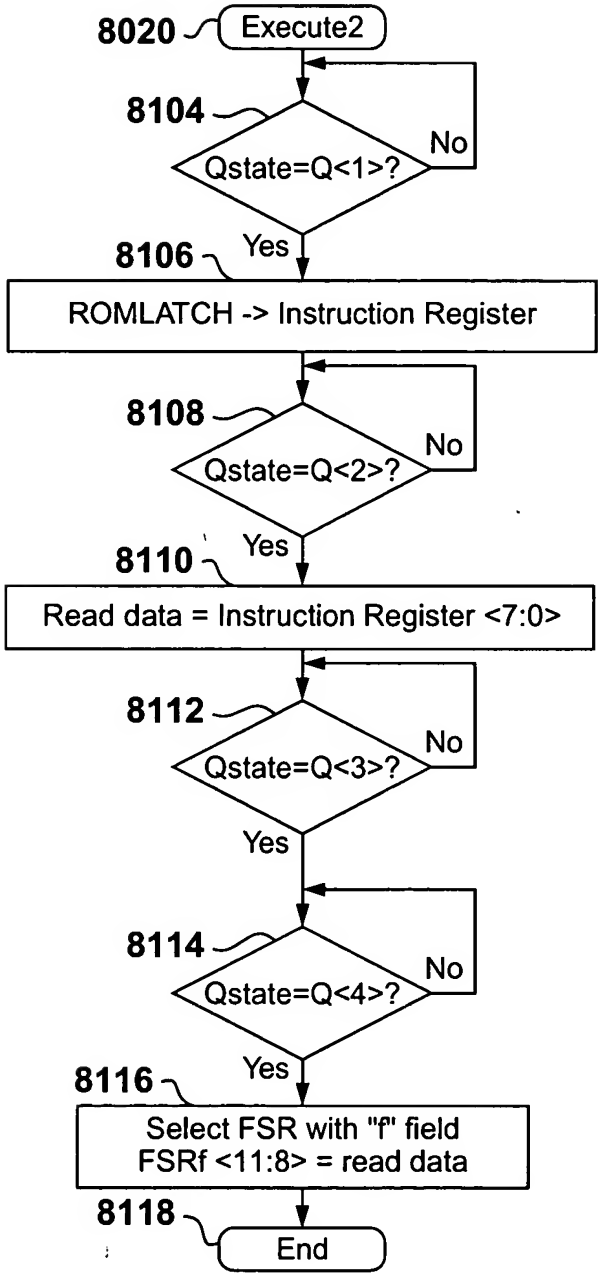

8020
Execute2
8104
Qstate=Q<1>?
No
Yes
8106
ROMLATCH -> Instruction Register
8108
Qstate=Q<2>?
No
Yes
8110
Read data = Instruction Register <7:0>
8112
Qstate=Q<3>?
No
Yes
8114
Qstate=Q<4>?
No
Yes
8116
Select FSR with "f" field
FSRf <11:8> = read data
8118
End

Figure 81

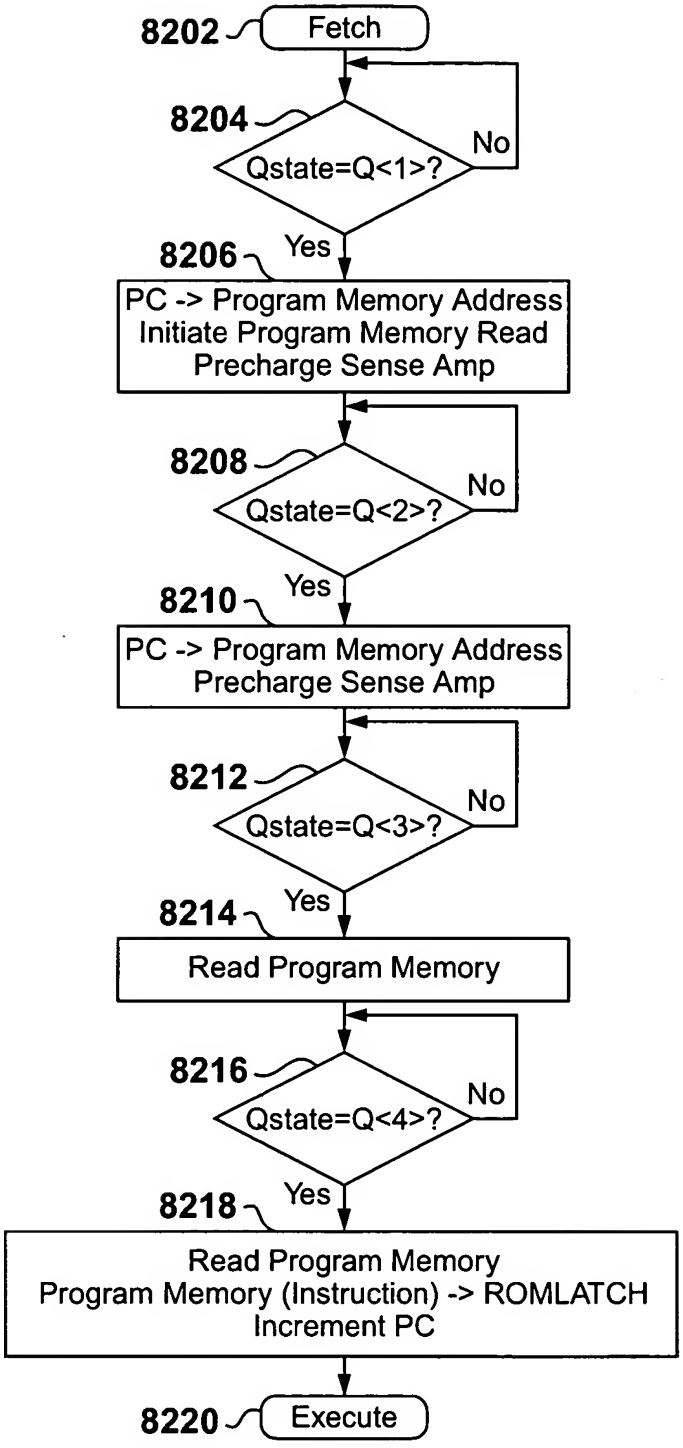

8202
Fetch
8204
Qstate=Q<1>?
No
Yes
8206
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
8208
Qstate=Q<2>?
No
Yes
8210
PC -> Program Memory Address
Precharge Sense Amp
8212
Qstate=Q<3>?
No
Yes
8214
Read Program Memory
8216
Qstate=Q<4>?
No
Yes
8218
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
8220
Execute

Figure 82

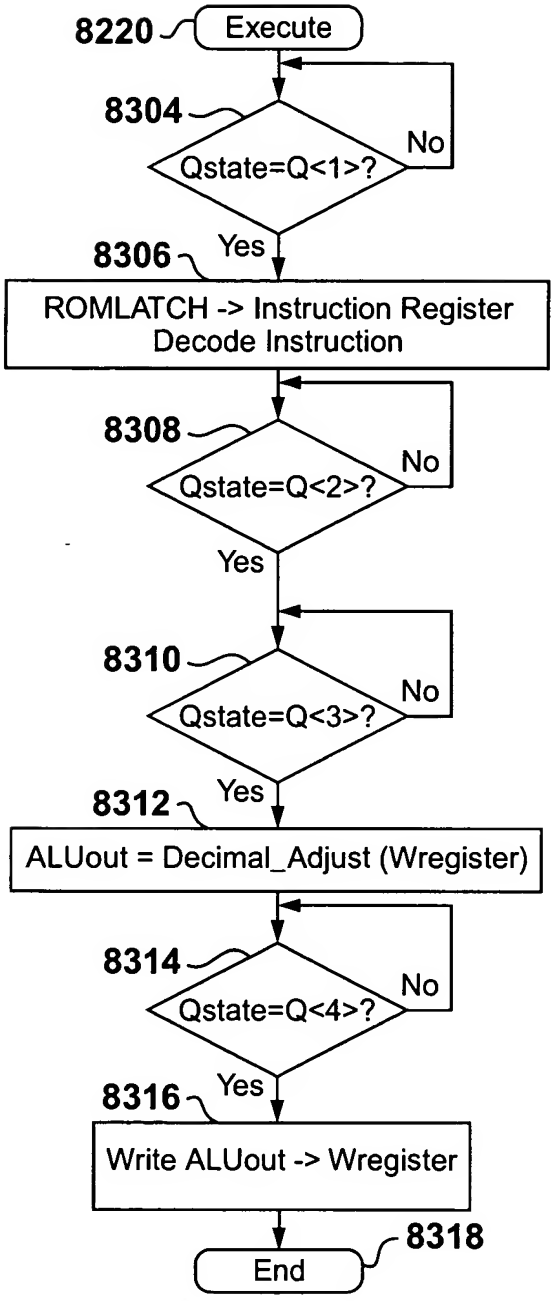

8220
Execute
8304
Qstate=Q<1>?
No
Yes
8306
ROMLATCH -> Instruction Register
Decode Instruction
8308
Qstate=Q<2>?
No
Yes
8310
Qstate=Q<3>?
No
Yes
8312
ALUout = Decimal_Adjust (Wregister)
8314
Qstate=Q<4>?
No
Yes
8316
Write ALUout -> Wregister
8318
End

Figure 83

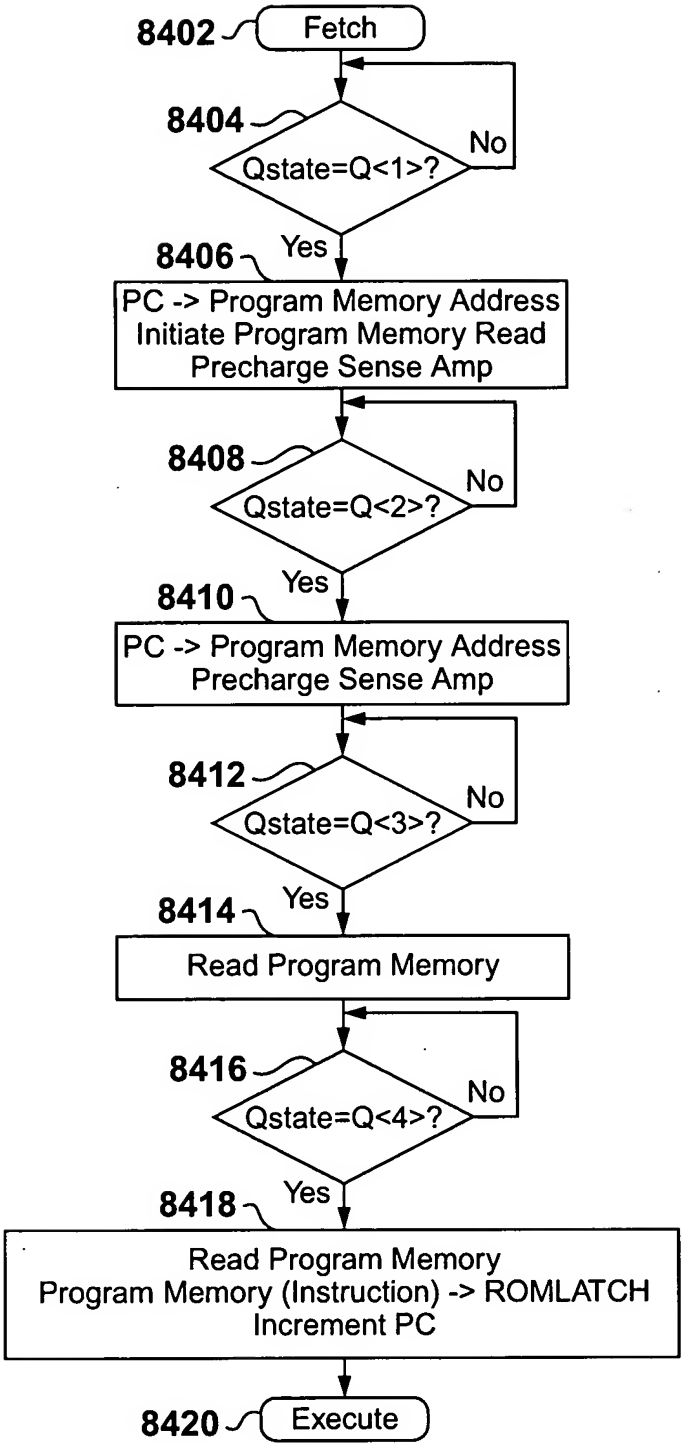

8402
Fetch
8404
Qstate=Q<1>?
No
Yes
8406
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
8408
Qstate=Q<2>?
No
Yes
8410
PC -> Program Memory Address
Precharge Sense Amp
8412
Qstate=Q<3>?
No
Yes
8414
Read Program Memory
8416
Qstate=Q<4>?
No
Yes
8418
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
8420
Execute

Figure 84

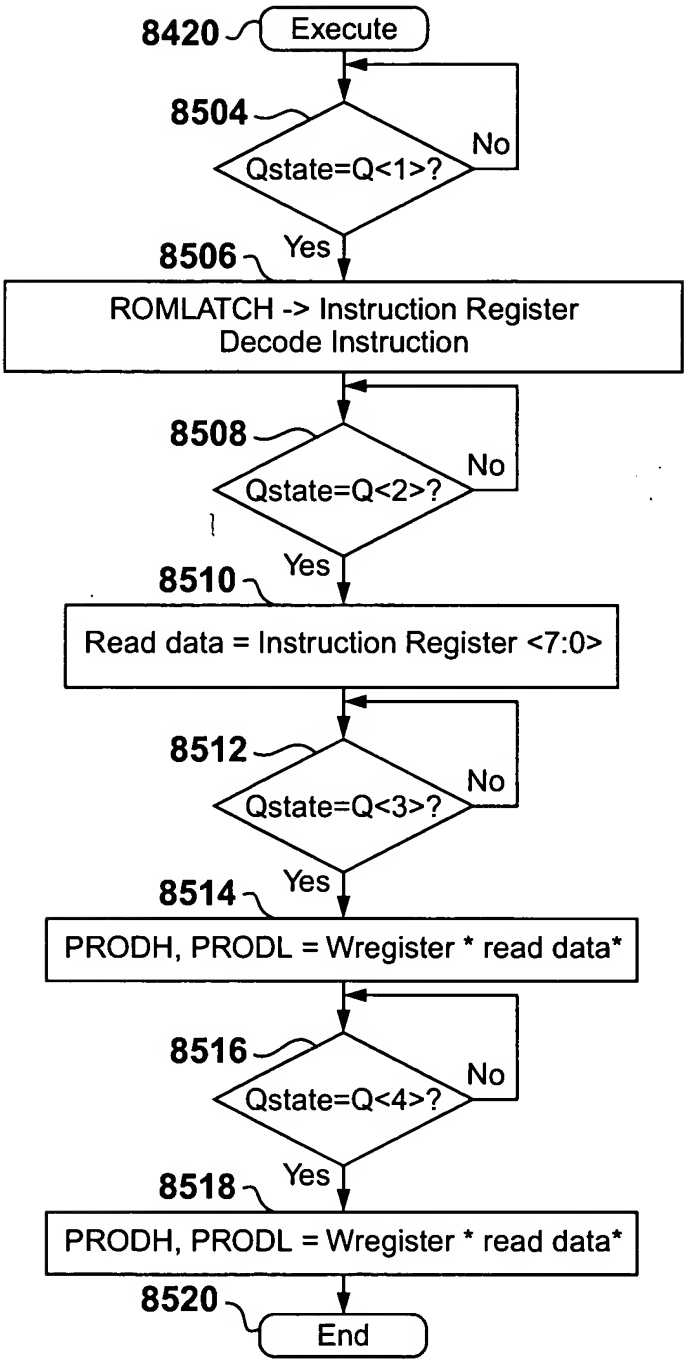

8420
Execute
8504
Qstate=Q<1>?
No
Yes
8506
ROMLATCH -> Instruction Register
Decode Instruction
8508
Qstate=Q<2>?
No
Yes
8510
Read data = Instruction Register <7:0>
8512
Qstate=Q<3>?
No
Yes
8514
PRODH, PRODL = Wregister * read data*
8516
Qstate=Q<4>?
No
Yes
8518
PRODH, PRODL = Wregister * read data*
8520
End

Figure 85

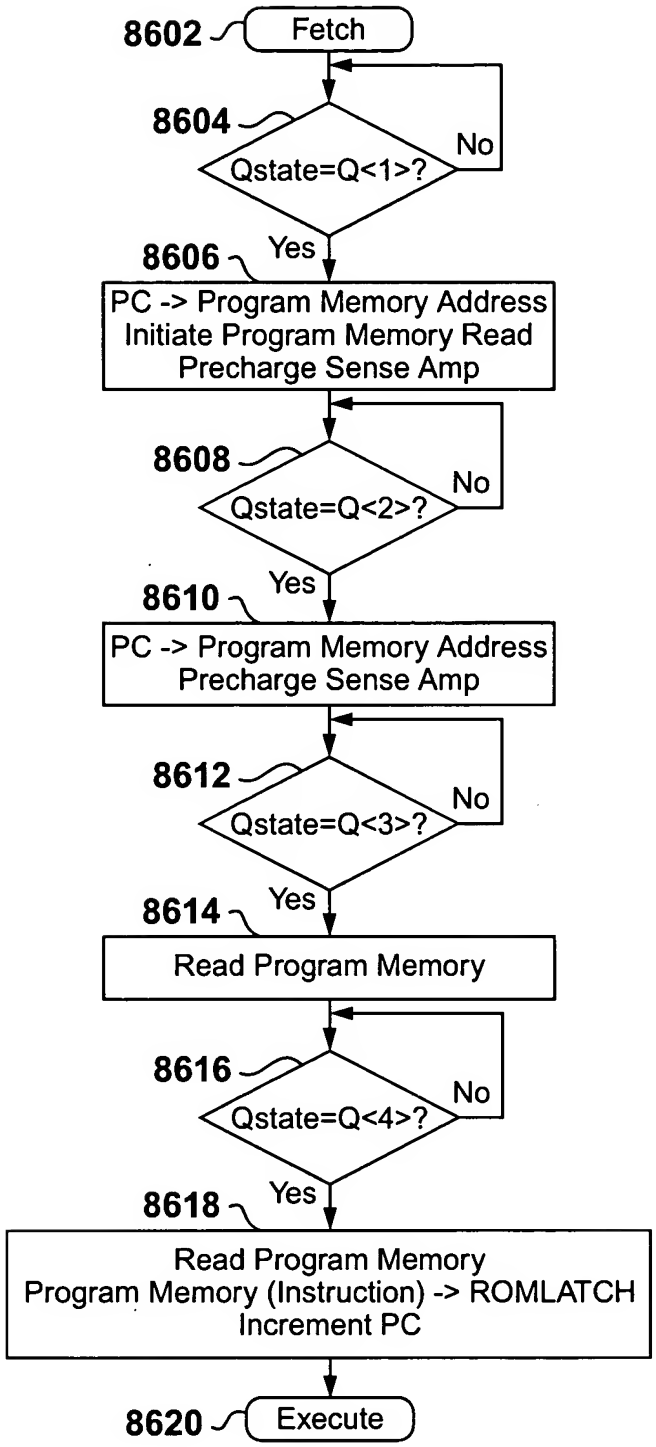

8602
Fetch
8604
Qstate=Q<1>?
No
Yes
8606
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
8608
Qstate=Q<2>?
No
Yes
8610
PC -> Program Memory Address
Precharge Sense Amp
8612
Qstate=Q<3>?
No
Yes
8614
Read Program Memory
8616
Qstate=Q<4>?
No
Yes
8618
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
8620
Execute

Figure 86

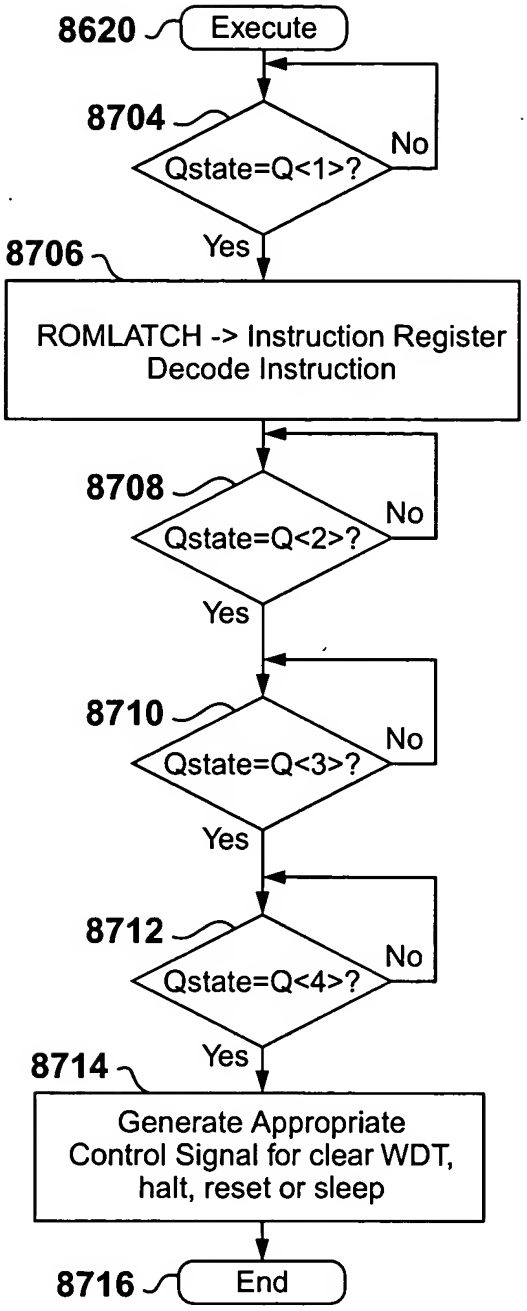

8620
Execute
8704
Qstate=Q<1>?
No
Yes
8706
ROMLATCH -> Instruction Register
Decode Instruction
8708
Qstate=Q<2>?
No
Yes
8710
Qstate=Q<3>?
No
Yes
8712
Qstate=Q<4>?
No
Yes
8714
Generate Appropriate
Control Signal for clear WDT,
halt, reset or sleep
8716
End

Figure 87

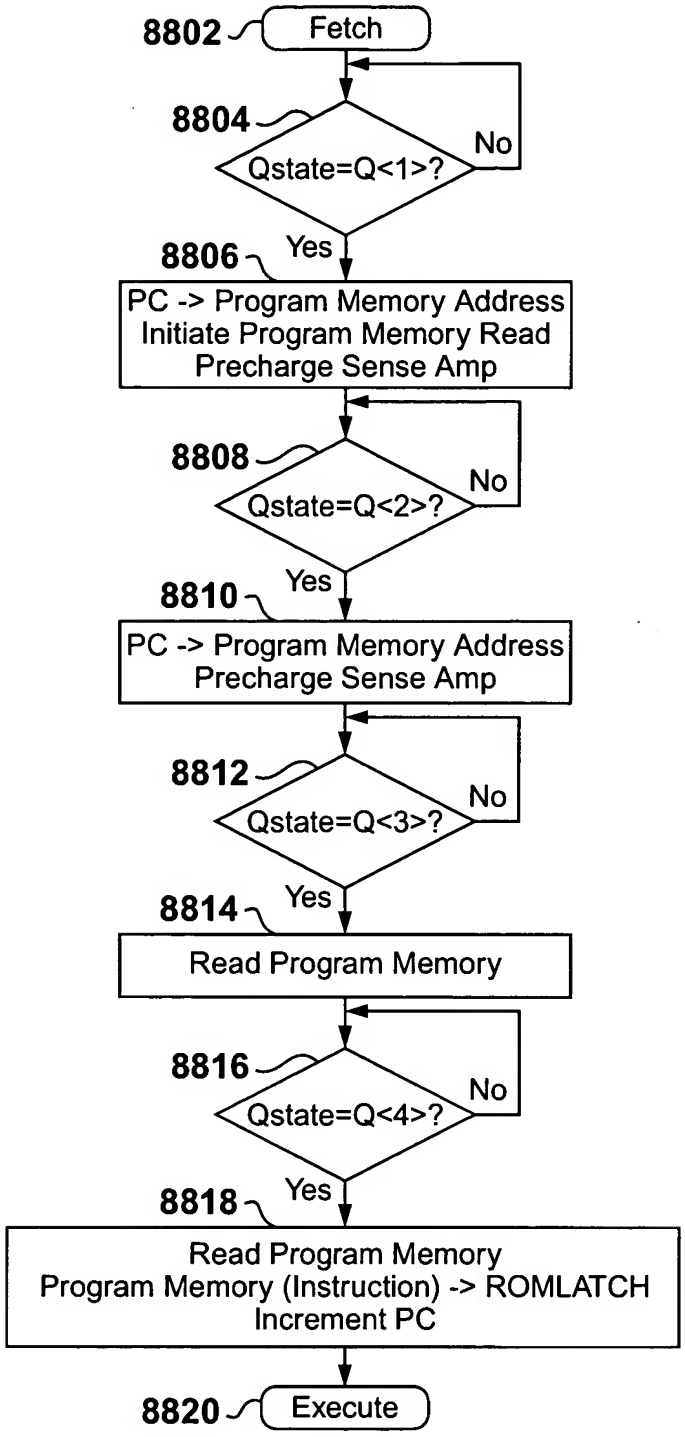

8802
Fetch
8804
Qstate=Q<1>?
No
Yes
8806
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
8808
Qstate=Q<2>?
No
Yes
8810
PC -> Program Memory Address
Precharge Sense Amp
8812
Qstate=Q<3>?
No
Yes
8814
Read Program Memory
8816
Qstate=Q<4>?
No
Yes
8818
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
8820
Execute

Figure 88

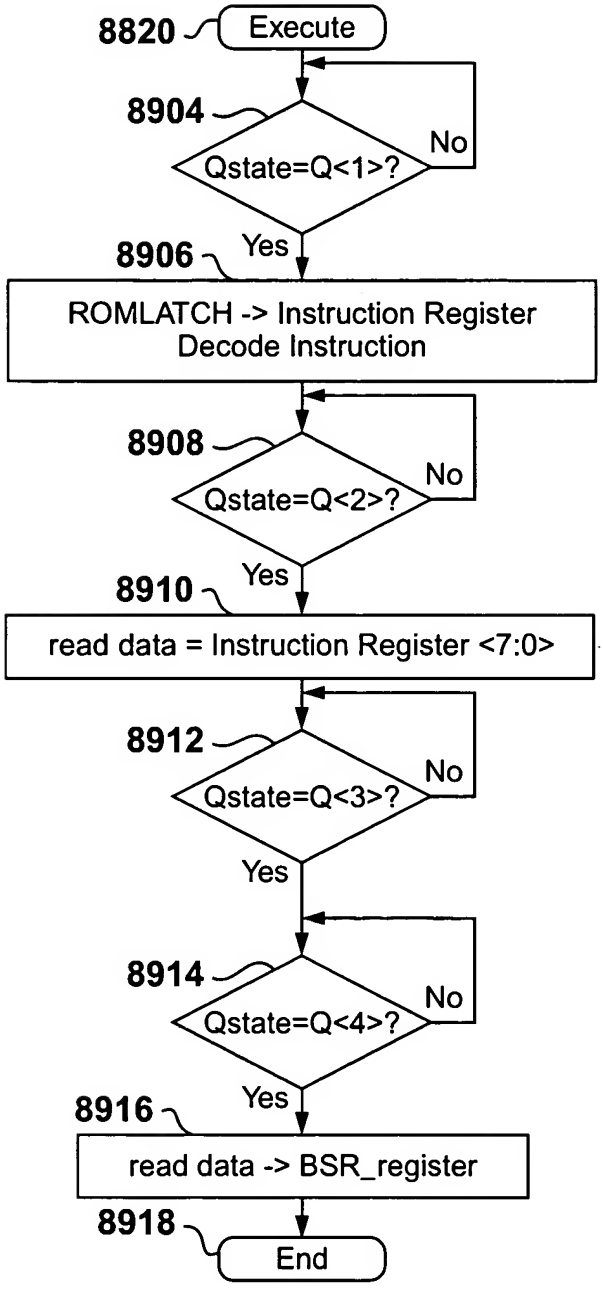
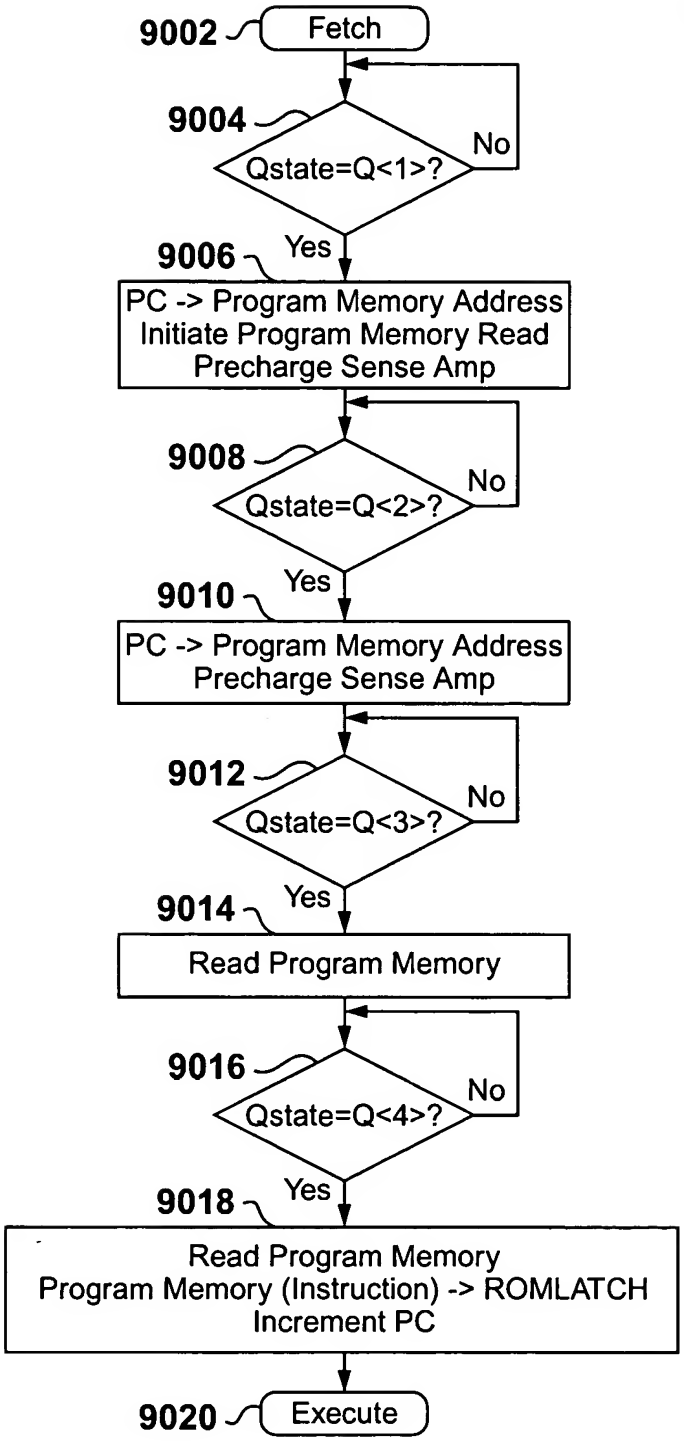

8820
Execute
8904
Qstate=Q<1>?
No
Yes
8906
ROMLATCH -> Instruction Register
Decode Instruction
8908
Qstate=Q<2>?
No
Yes
8910
read data = Instruction Register <7:0>
8912
Qstate=Q<3>?
No
Yes
8914
Qstate=Q<4>?
No
Yes
8916
read data -> BSR_register
8918
End

Figure 89



Valid For:
BC
BN
BNC
BNN
BNV
BNZ
BV
BZ

Figure 90

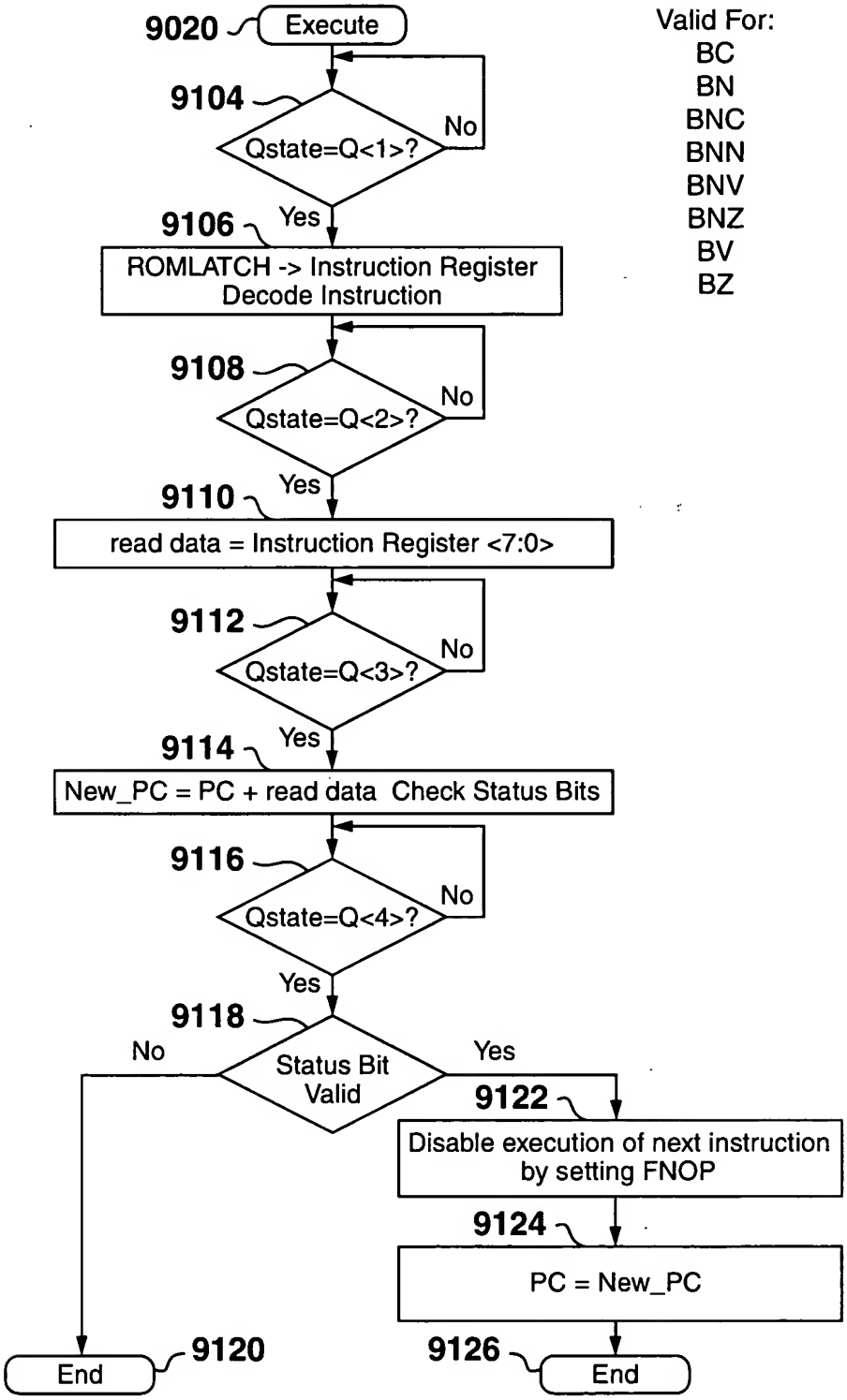

9020
Execute
9104
Qstate=Q<1>?
No
Yes
9106
ROMLATCH -> Instruction Register
Decode Instruction
9108
Qstate=Q<2>?
No
Yes
9110
read data = Instruction Register <7:0>
9112
Qstate=Q<3>?
No
Yes
9114
New_PC = PC + read data Check Status Bits
9116
Qstate=Q<4>?
No
Yes
9118
Status Bit
Valid
No
Yes
9122
Disable execution of next instruction
by setting FNOP
9124
PC = New_PC
9120
End
9126
End
Valid For:
BC
BN
BNC
BNN
BNV
BNZ
BV
BZ

Figure 91

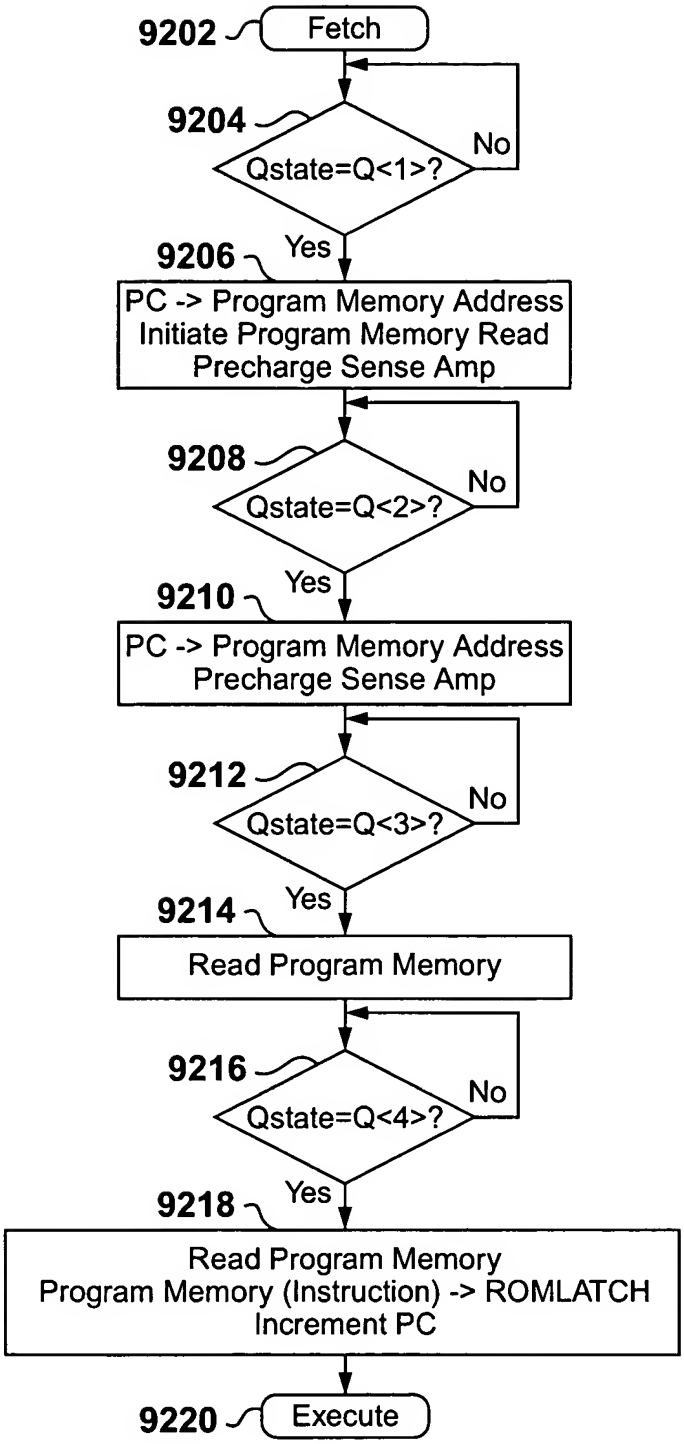

9202
Fetch
9204
Qstate=Q<1>?
No
Yes
9206
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
9208
Qstate=Q<2>?
No
Yes
9210
PC -> Program Memory Address
Precharge Sense Amp
9212
Qstate=Q<3>?
No
Yes
9214
Read Program Memory
9216
Qstate=Q<4>?
No
Yes
9218
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
9220
Execute

Figure 92

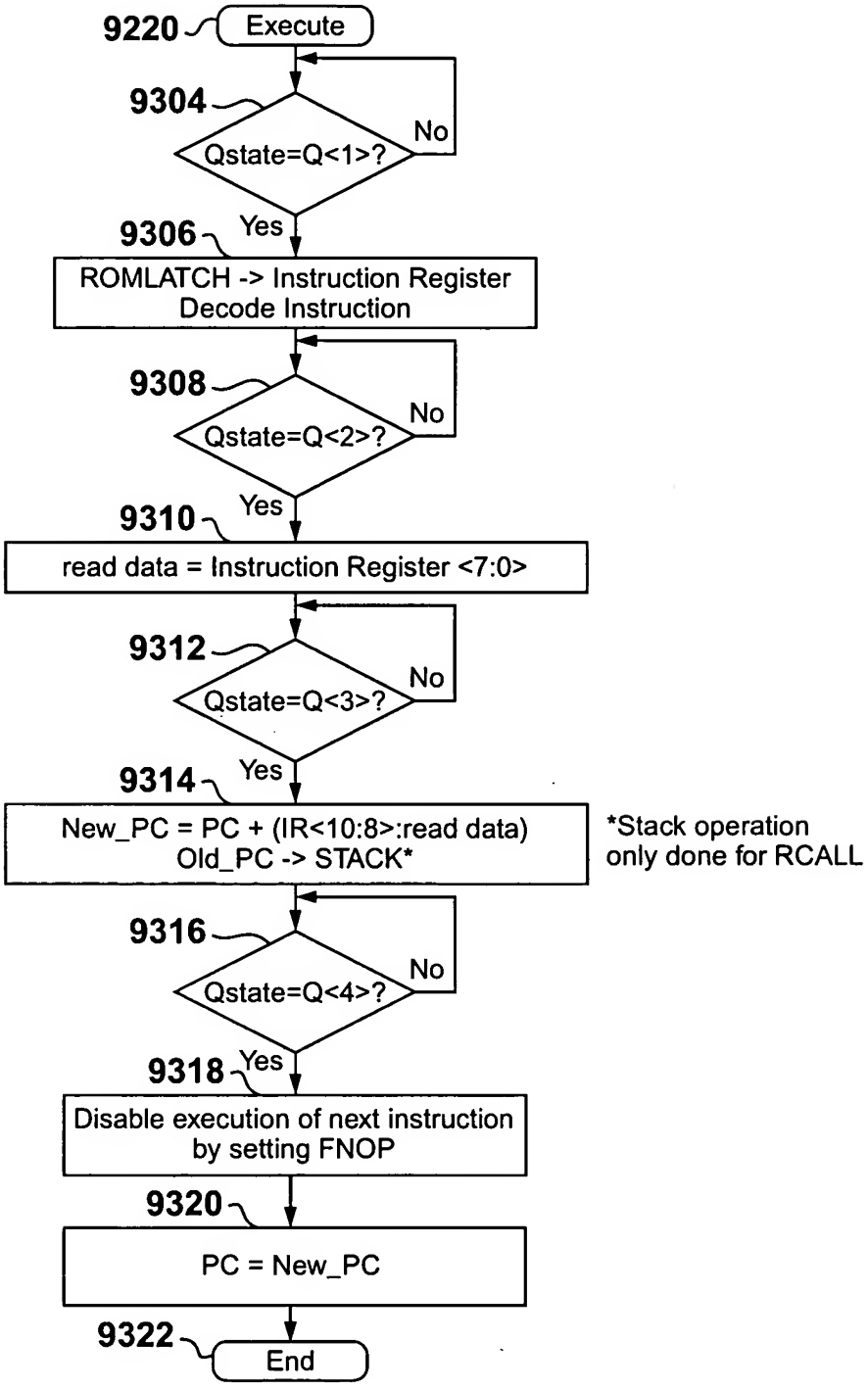

9220
Execute
9304
Qstate=Q<1>?
No
Yes
9306
ROMLATCH -> Instruction Register
Decode Instruction
9308
Qstate=Q<2>?
No
Yes
9310
read data = Instruction Register <7:0>
9312
Qstate=Q<3>?
No
Yes
9314
New_PC = PC + (IR<10:8>:read data)
Old_PC -> STACK*
*Stack operation
only done for RCALL
9316
Qstate=Q<4>?
No
Yes
9318
Disable execution of next instruction
by setting FNOP
9320
PC = New_PC
9322
End

Figure 93

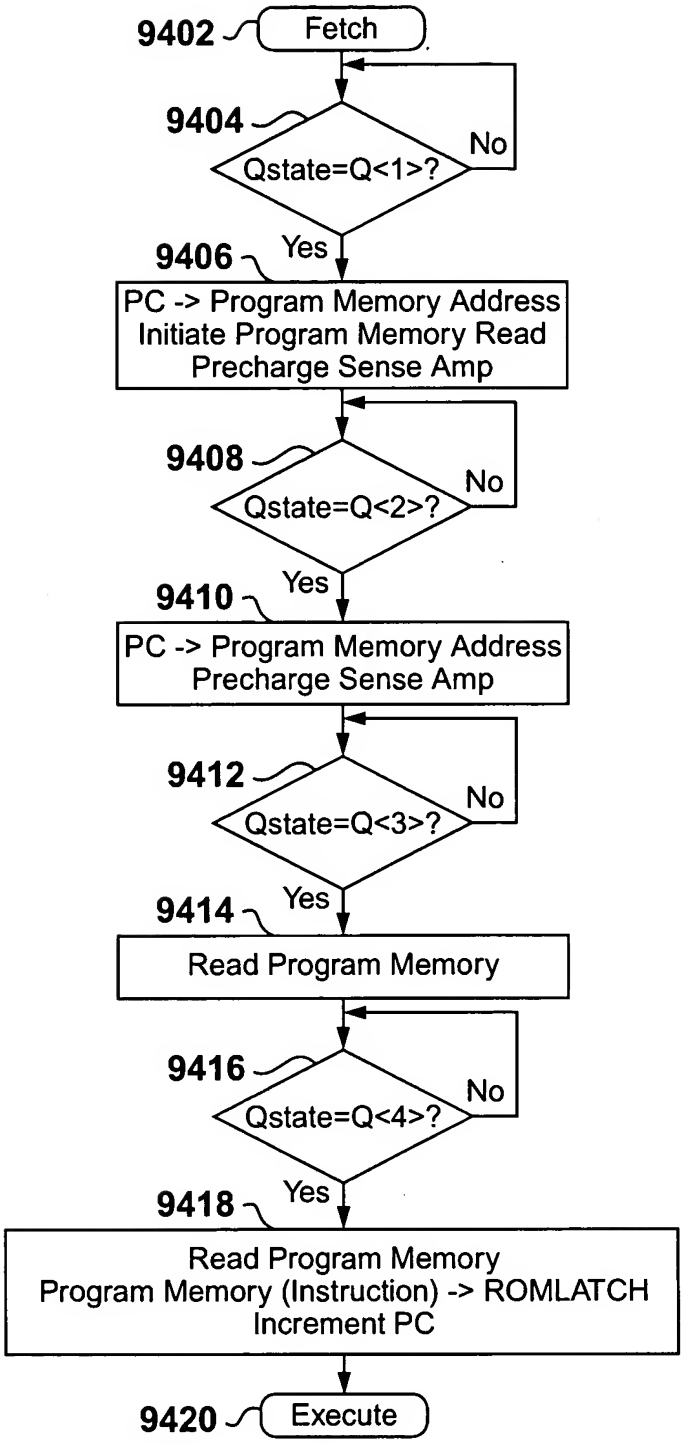

9402
Fetch
9404
Qstate=Q<1>?
No
Yes
9406
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
9408
Qstate=Q<2>?
No
Yes
9410
PC -> Program Memory Address
Precharge Sense Amp
9412
Qstate=Q<3>?
No
Yes
9414
Read Program Memory
9416
Qstate=Q<4>?
No
Yes
9418
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
9420
Execute

Figure 94

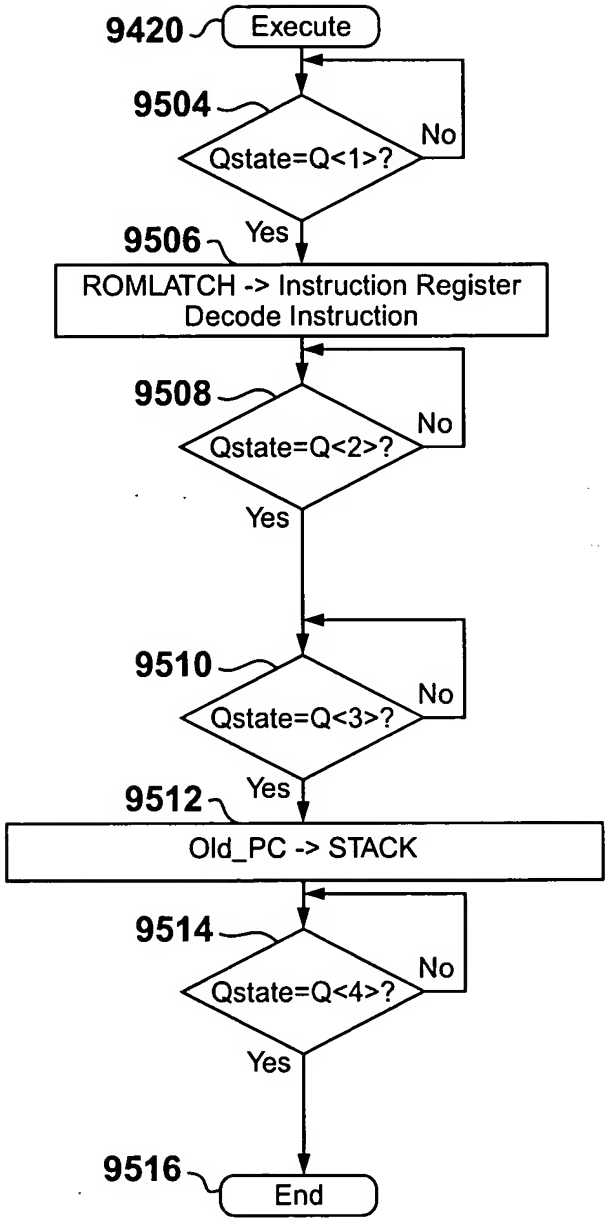

9420
Execute
9504
Qstate=Q<1>?
No
Yes
9506
ROMLATCH -> Instruction Register
Decode Instruction
9508
Qstate=Q<2>?
No
Yes
9510
Qstate=Q<3>?
No
Yes
9512
Old_PC -> STACK
9514
Qstate=Q<4>?
No
Yes
9516
End

Figure 95

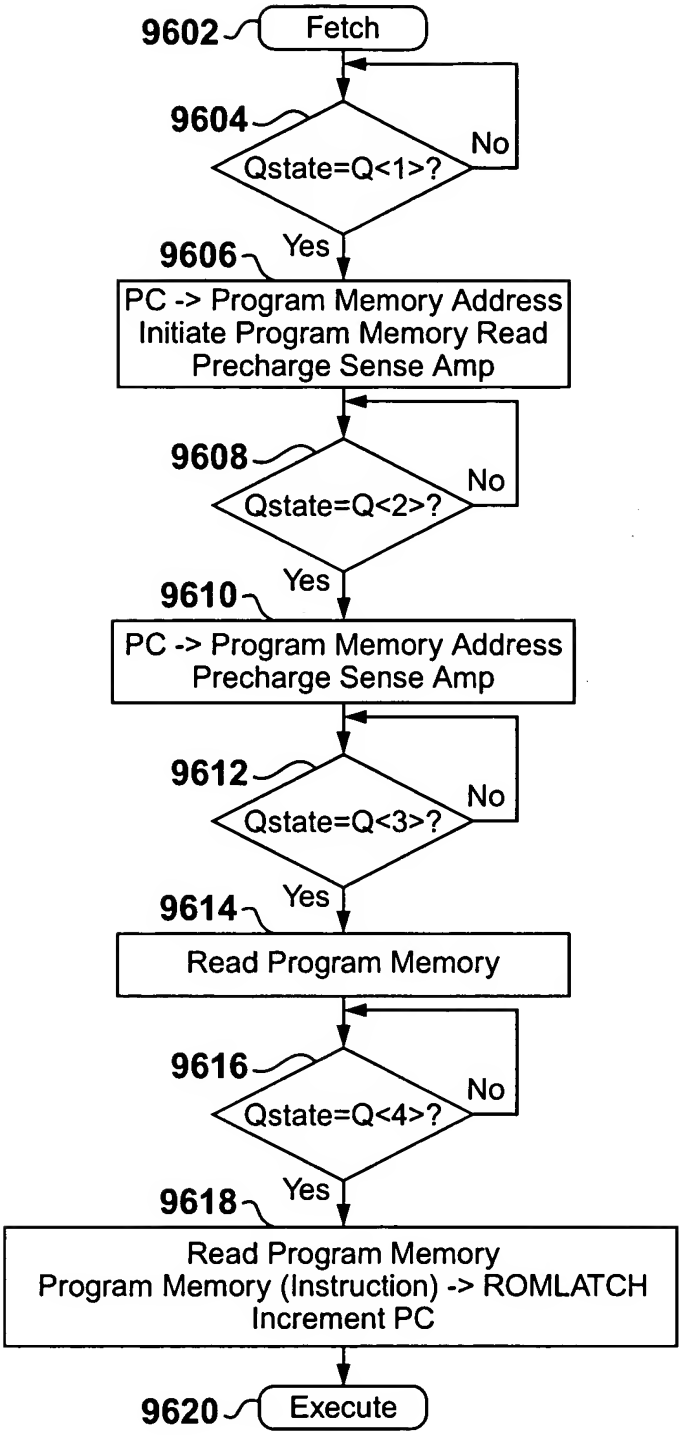

9602
Fetch
9604
Qstate=Q<1>?
No
Yes
9606
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
9608
Qstate=Q<2>?
No
Yes
9610
PC -> Program Memory Address
Precharge Sense Amp
9612
Qstate=Q<3>?
No
Yes
9614
Read Program Memory
9616
Qstate=Q<4>?
No
Yes
9618
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
9620
Execute

Figure 96

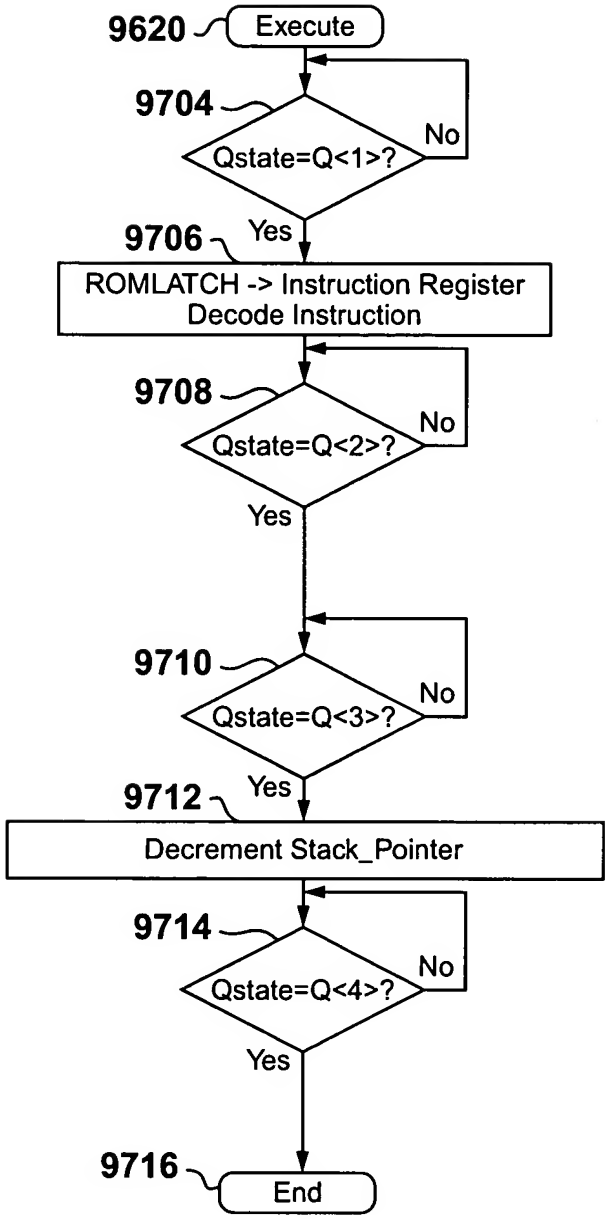

9620
Execute
9704
Qstate=Q<1>?
No
Yes
9706
ROMLATCH -> Instruction Register
Decode Instruction
9708
Qstate=Q<2>?
No
Yes
9710
Qstate=Q<3>?
No
Yes
9712
Decrement Stack_Pointer
9714
Qstate=Q<4>?
No
Yes
9716
End

Figure 97

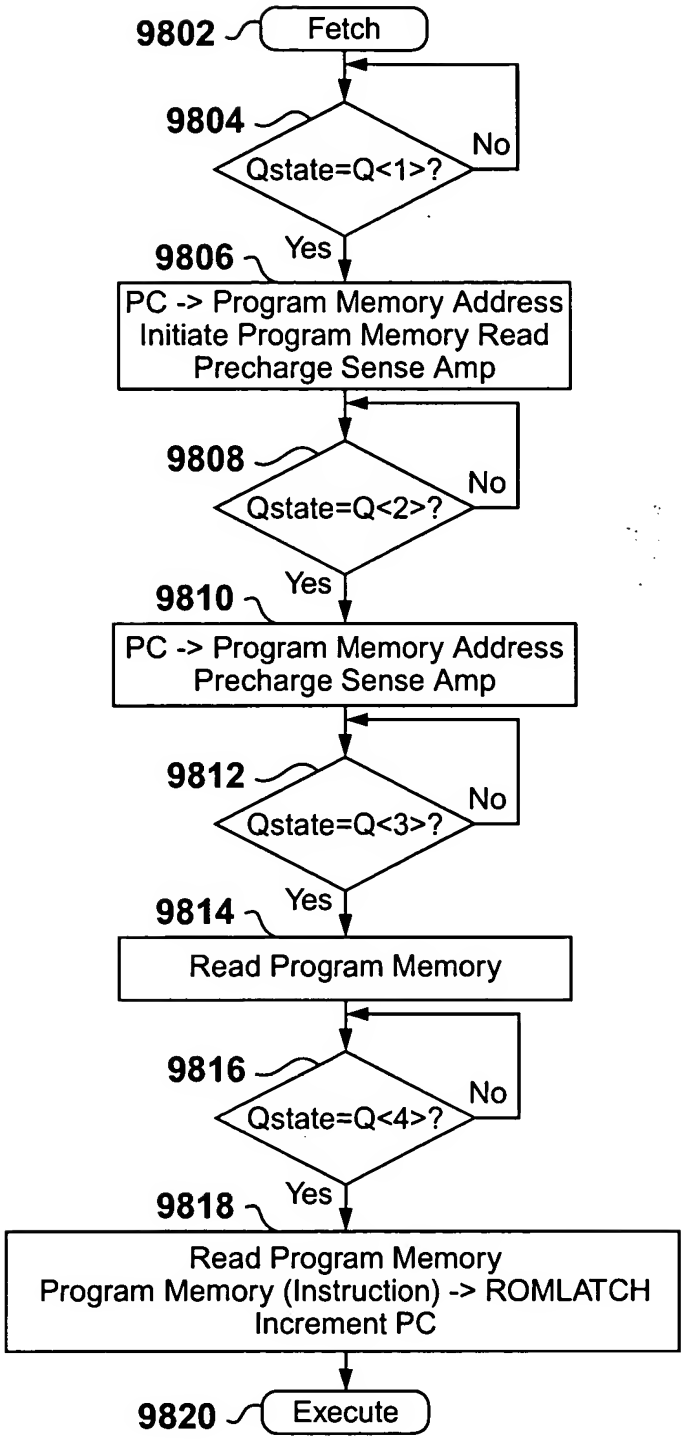

9802
Fetch
9804
Qstate=Q<1>?
No
Yes
9806
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
9808
Qstate=Q<2>?
No
Yes
9810
PC -> Program Memory Address
Precharge Sense Amp
9812
Qstate=Q<3>?
No
Yes
9814
Read Program Memory
9816
Qstate=Q<4>?
No
Yes
9818
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
9820
Execute

Figure 98

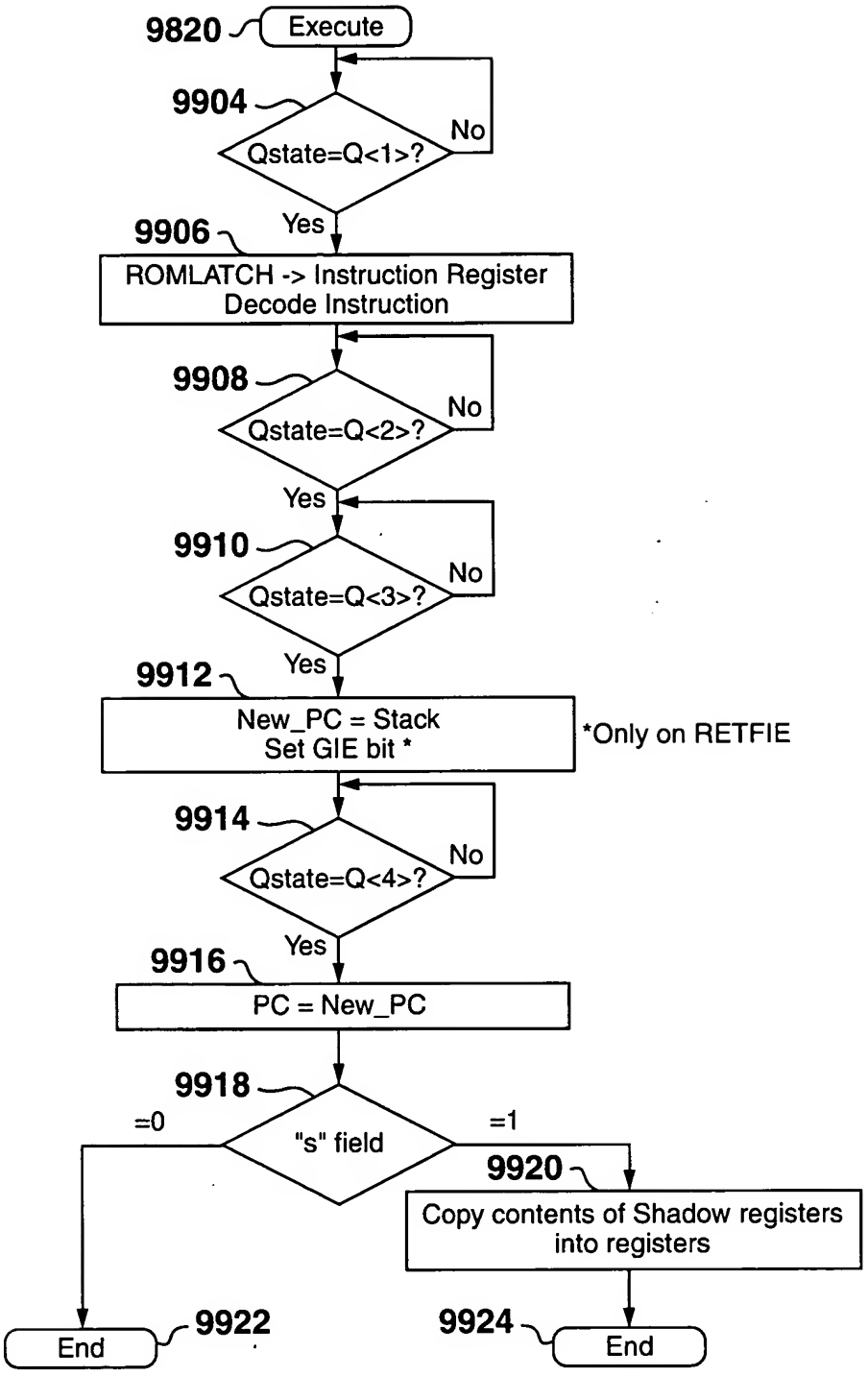

9820
Execute
9904
Qstate=Q<1>?
No
Yes
9906
ROMLATCH -> Instruction Register
Decode Instruction
9908
Qstate=Q<2>?
No
Yes
9910
Qstate=Q<3>?
No
Yes
9912
New_PC = Stack
Set GIE bit *
*Only on RETFIE
9914
Qstate=Q<4>?
No
Yes
9916
PC = New_PC
9918
"s" field
=0
=1
9920
Copy contents of Shadow registers
into registers
9922
End
9924
End

Figure 99

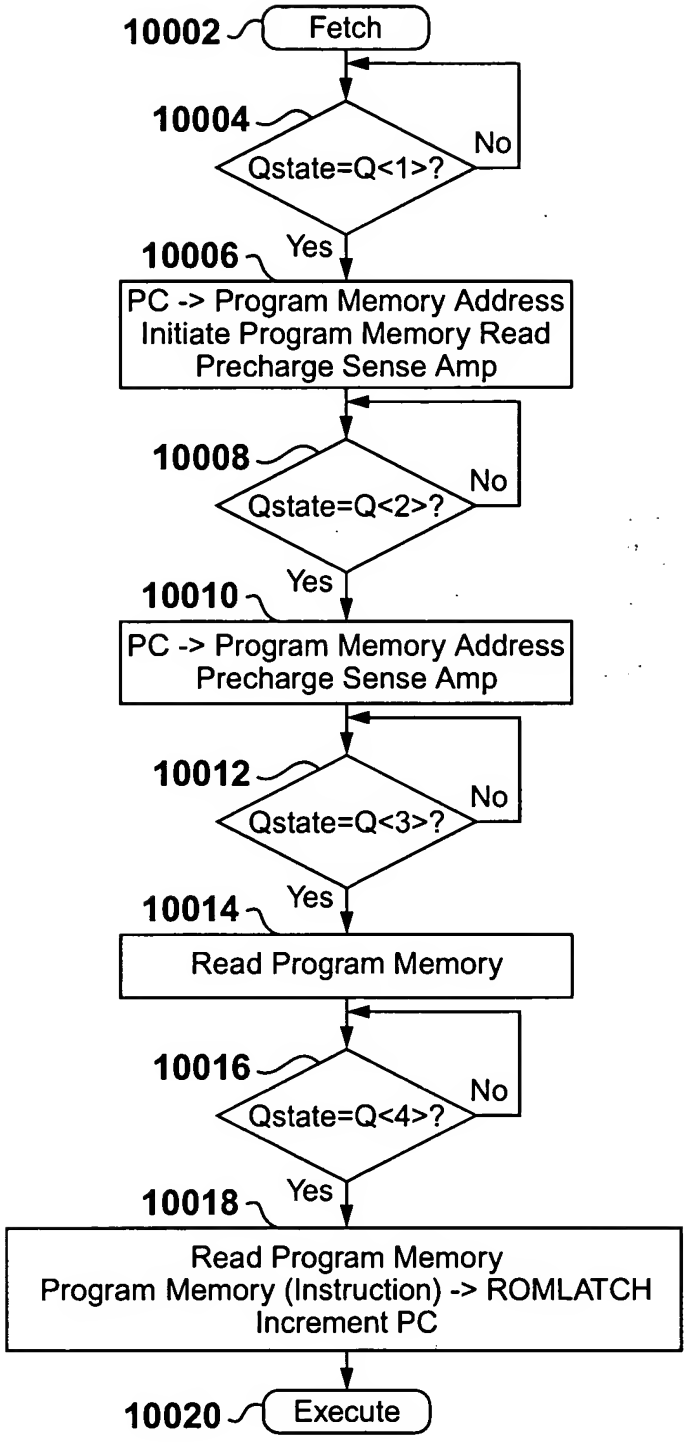

10002
Fetch
10004
Qstate=Q<1>?
No
Yes
10006
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
10008
Qstate=Q<2>?
No
Yes
10010
PC -> Program Memory Address
Precharge Sense Amp
10012
Qstate=Q<3>?
No
Yes
10014
Read Program Memory
10016
Qstate=Q<4>?
No
Yes
10018
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
10020
Execute

Figure 100

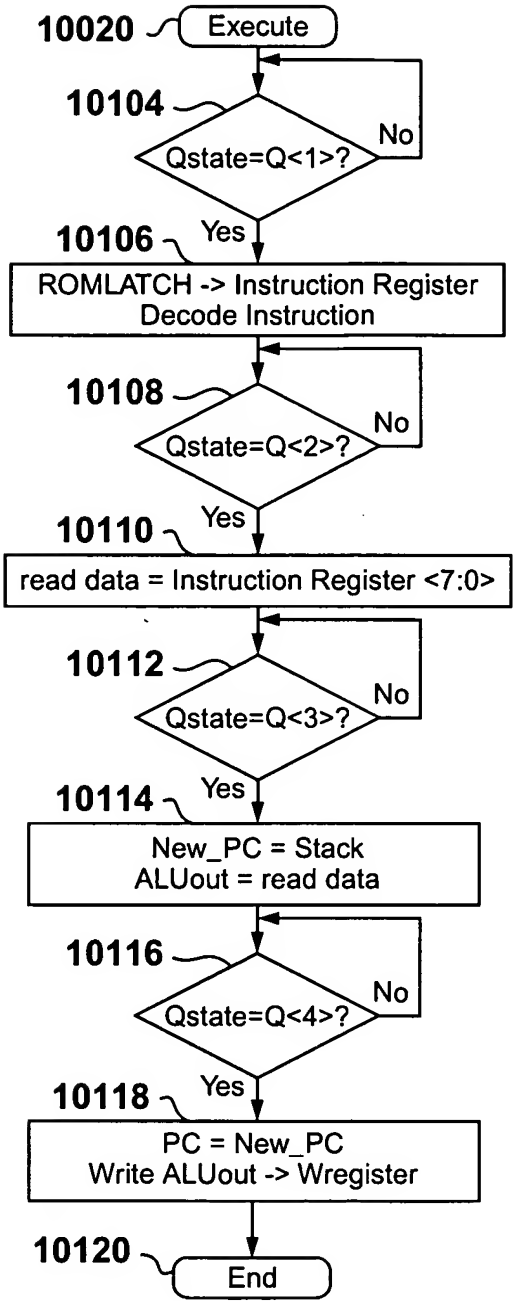

10020
Execute
10104
Qstate=Q<1>?
No
Yes
10106
ROMLATCH -> Instruction Register
Decode Instruction
10108
Qstate=Q<2>?
No
Yes
10110
read data = Instruction Register <7:0>
10112
Qstate=Q<3>?
No
Yes
10114
New_PC = Stack
ALUout = read data
10116
Qstate=Q<4>?
No
Yes
10118
PC = New_PC
Write ALUout -> Wregister
10120
End

Figure 101

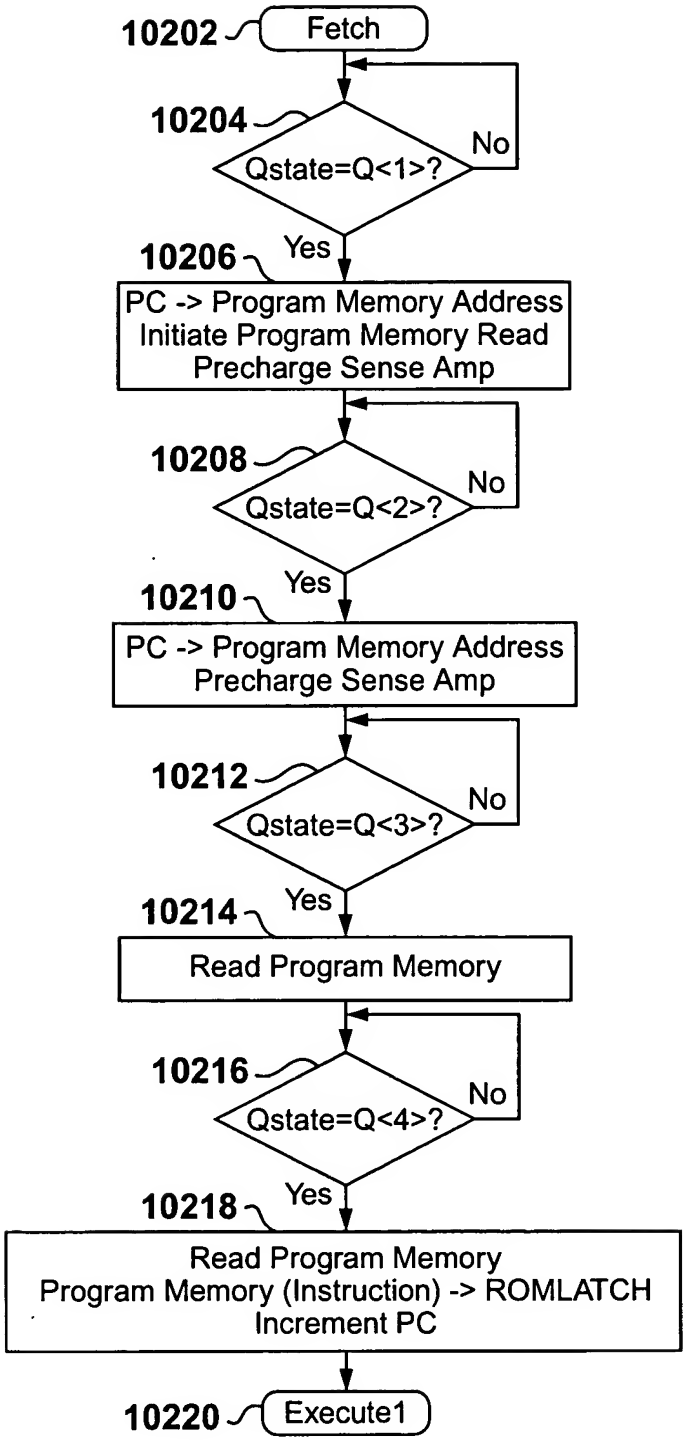

10202
Fetch
10204
Qstate=Q<1>?
No
Yes
10206
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
10208
Qstate=Q<2>?
No
Yes
10210
PC -> Program Memory Address
Precharge Sense Amp
10212
Qstate=Q<3>?
No
Yes
10214
Read Program Memory
10216
Qstate=Q<4>?
No
Yes
10218
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
10220
Execute1

Figure 102

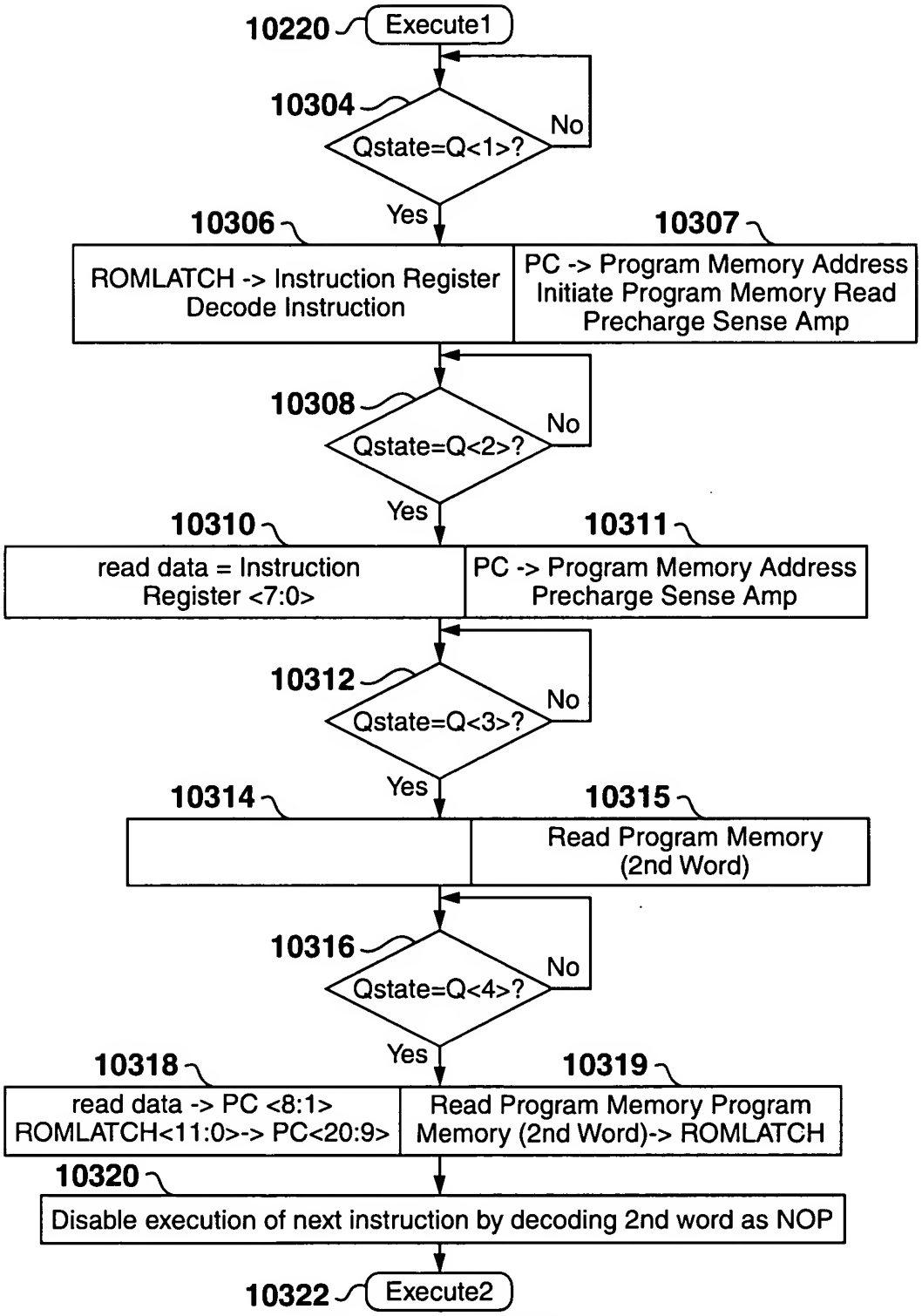

10220
Execute1
10304
Qstate=Q<1>?
No
Yes
10306
ROMLATCH -> Instruction Register
Decode Instruction
10307
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
10308
Qstate=Q<2>?
No
Yes
10310
read data = Instruction
Register <7:0>
10311
PC -> Program Memory Address
Precharge Sense Amp
10312
Qstate=Q<3>?
No
Yes
10314
10315
Read Program Memory
(2nd Word)
10316
Qstate=Q<4>?
No
Yes
10318
read data -> PC <8:1>
ROMLATCH<11:0>-> PC<20:9>
10319
Read Program Memory Program
Memory (2nd Word)-> ROMLATCH
10320
Disable execution of next instruction by decoding 2nd word as NOP
10322
Execute2

Figure 103

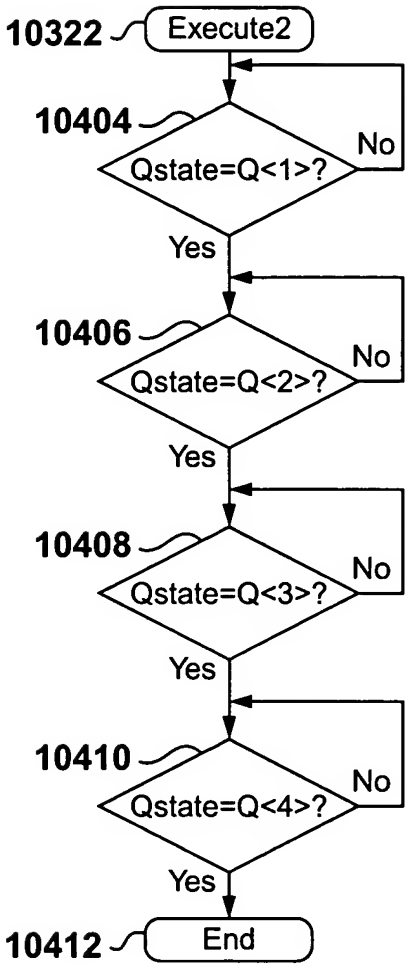

10322
Execute2
10404
Qstate=Q<1>?
No
Yes
10406
Qstate=Q<2>?
No
Yes
10408
Qstate=Q<3>?
No
Yes
10410
Qstate=Q<4>?
No
Yes
10412
End

Figure 104

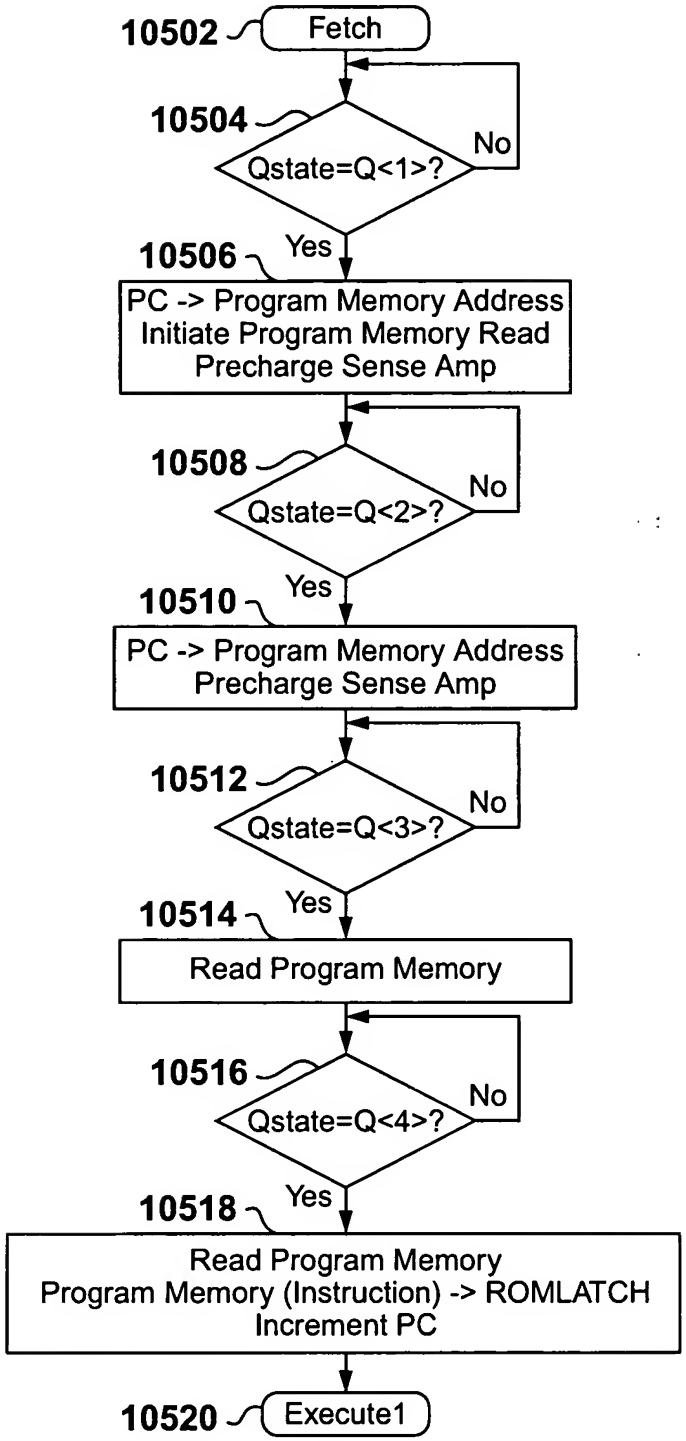

10502
Fetch
10504
Qstate=Q<1>?
No
Yes
10506
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
10508
Qstate=Q<2>?
No
Yes
10510
PC -> Program Memory Address
Precharge Sense Amp
10512
Qstate=Q<3>?
No
Yes
10514
Read Program Memory
10516
Qstate=Q<4>?
No
Yes
10518
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
10520
Execute1

Figure 105

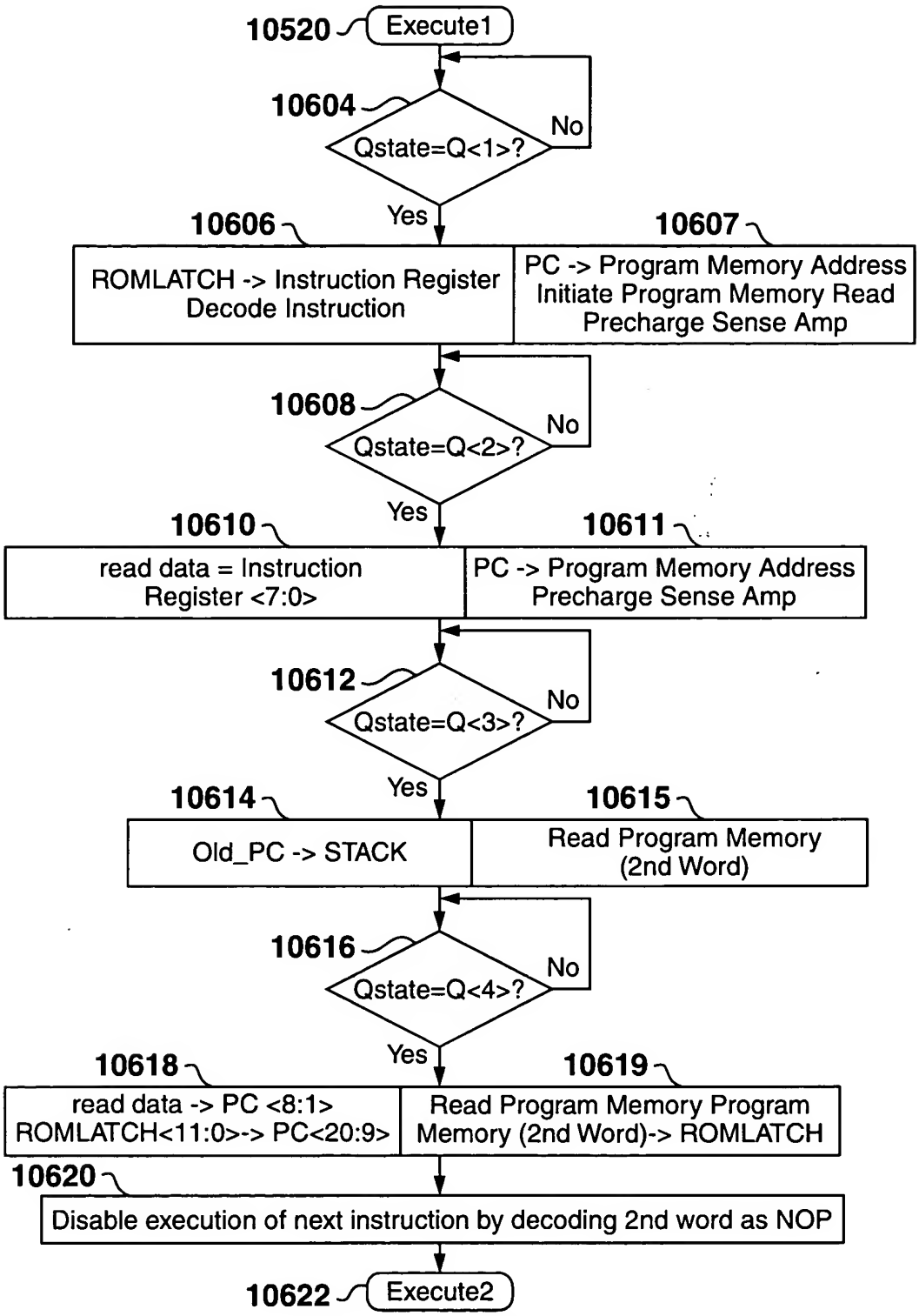

10520
Execute1
10604
Qstate=Q<1>?
No
Yes
10606
ROMLATCH -> Instruction Register
Decode Instruction
10607
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
10608
Qstate=Q<2>?
No
Yes
10610
read data = Instruction
Register <7:0>
10611
PC -> Program Memory Address
Precharge Sense Amp
10612
Qstate=Q<3>?
No
Yes
10614
Old_PC -> STACK
10615
Read Program Memory
(2nd Word)
10616
Qstate=Q<4>?
No
Yes
10618
read data -> PC <8:1>
ROMLATCH<11:0>-> PC<20:9>
10619
Read Program Memory Program
Memory (2nd Word)-> ROMLATCH
10620
Disable execution of next instruction by decoding 2nd word as NOP
10622
Execute2

Figure 106

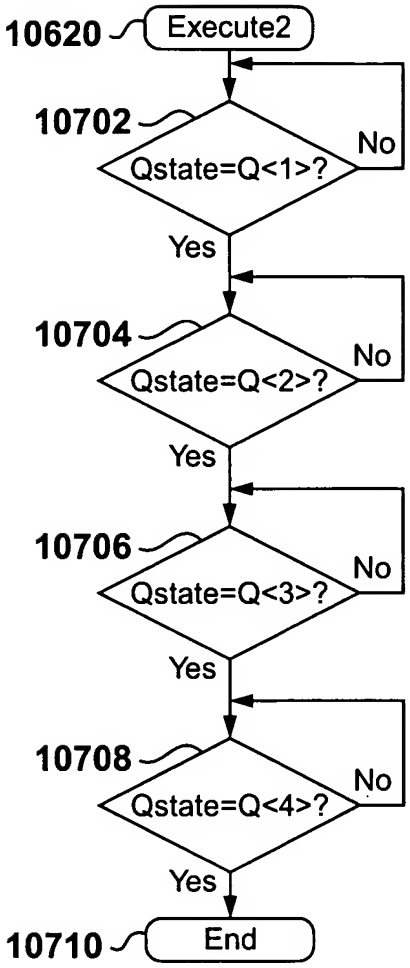

10620
Execute2
10702
Qstate=Q<1>?
No
Yes
10704
Qstate=Q<2>?
No
Yes
10706
Qstate=Q<3>?
No
Yes
10708
Qstate=Q<4>?
No
Yes
10710
End

Figure 107

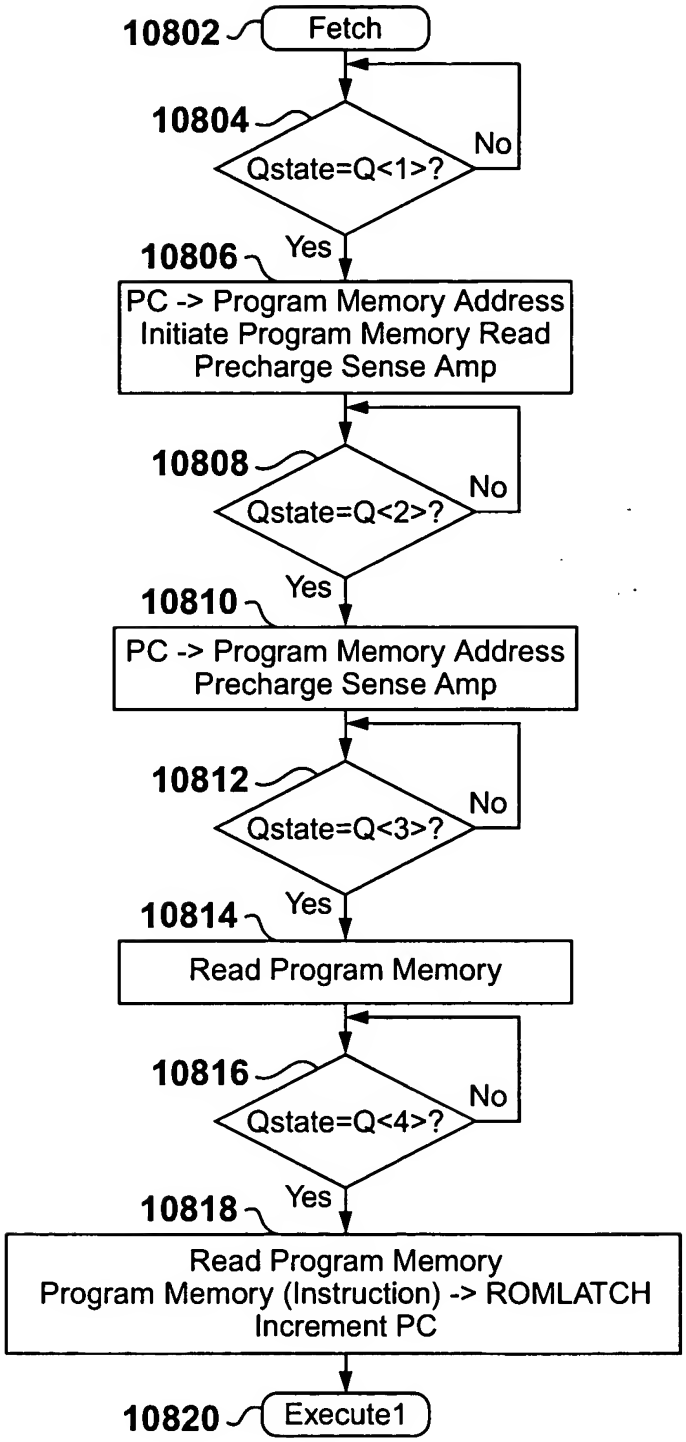

10802
Fetch
10804
Qstate=Q<1>?
No
Yes
10806
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
10808
Qstate=Q<2>?
No
Yes
10810
PC -> Program Memory Address
Precharge Sense Amp
10812
Qstate=Q<3>?
No
Yes
10814
Read Program Memory
10816
Qstate=Q<4>?
No
Yes
10818
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
10820
Execute1

Figure 108

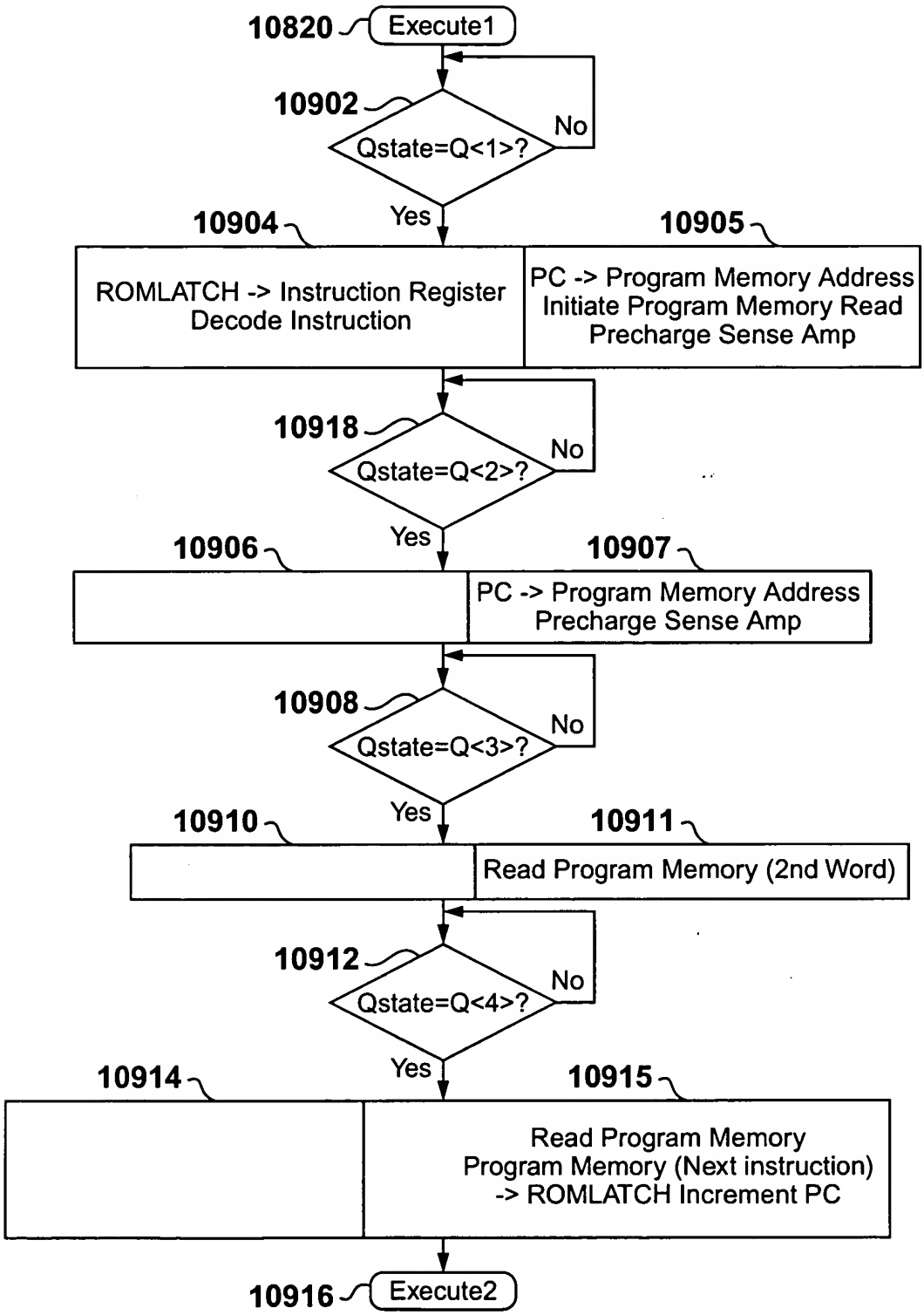

10820
Execute1
10902
Qstate=Q<1>?
No
Yes
10904
10905
ROMLATCH -> Instruction Register
Decode Instruction
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
10918
Qstate=Q<2>?
No
Yes
10906
10907
PC -> Program Memory Address
Precharge Sense Amp
10908
Qstate=Q<3>?
No
Yes
10910
10911
Read Program Memory (2nd Word)
10912
Qstate=Q<4>?
No
Yes
10914
10915
Read Program Memory
Program Memory (Next instruction)
-> ROMLATCH Increment PC
10916
Execute2

Figure 109

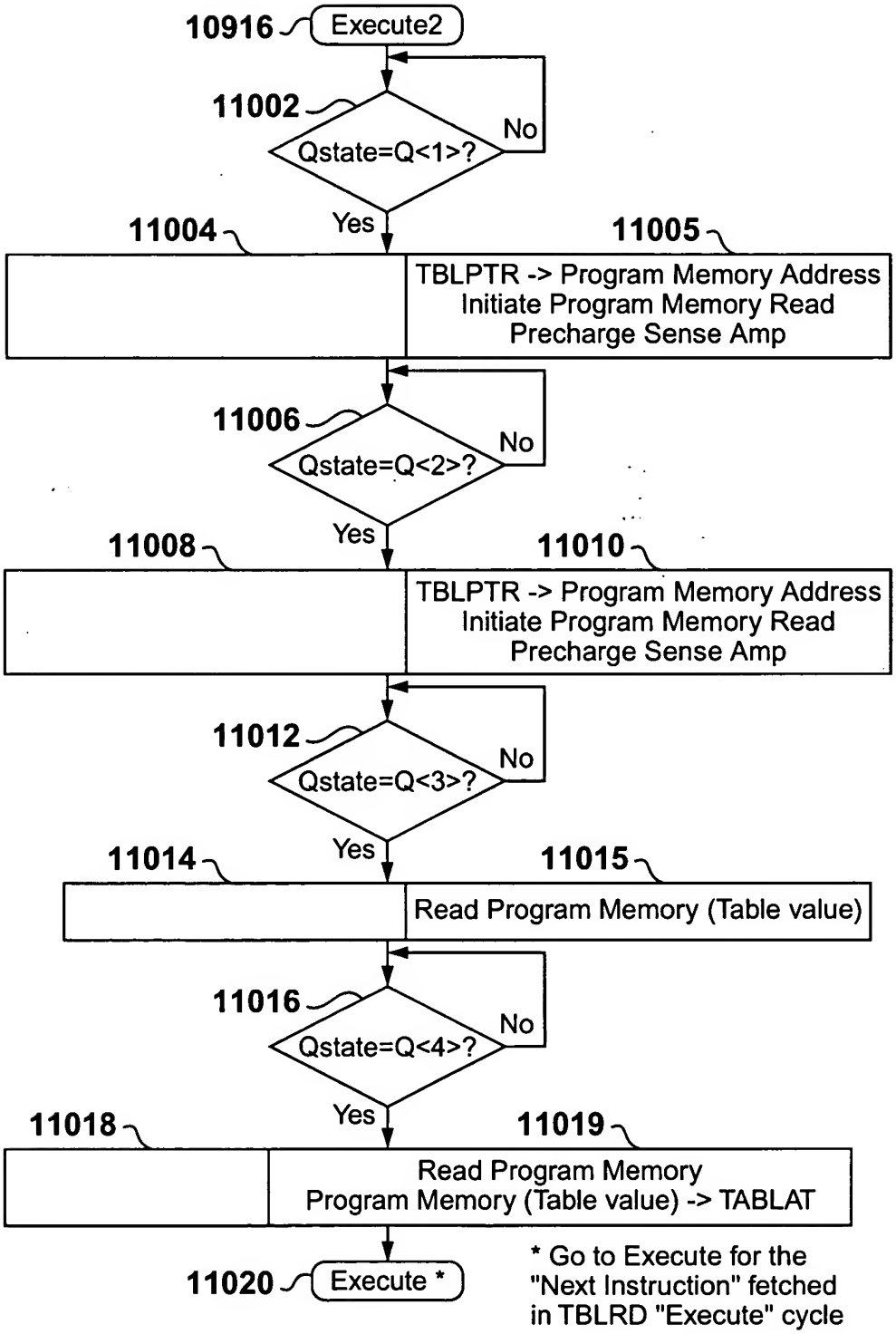

10916
Execute2
11002
Qstate=Q<1>?
No
Yes
11004
11005
TBLPTR -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
11006
Qstate=Q<2>?
No
Yes
11008
11010
TBLPTR -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
11012
Qstate=Q<3>?
No
Yes
11014
11015
Read Program Memory (Table value)
11016
Qstate=Q<4>?
No
Yes
11018
11019
Read Program Memory
Program Memory (Table value) -> TABLAT
11020
Execute *
* Go to Execute for the
"Next Instruction" fetched
in TBLRD "Execute" cycle

Figure 110

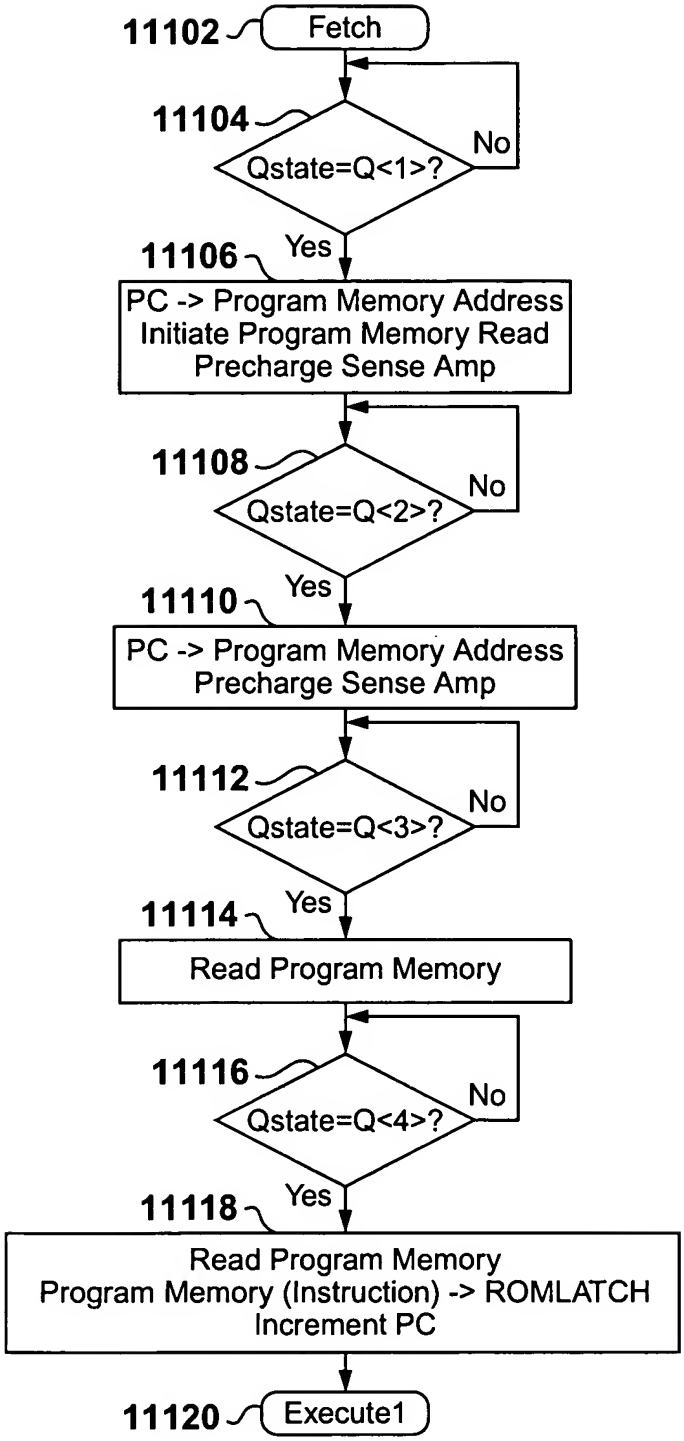

11102
Fetch
11104
Qstate=Q<1>?
No
Yes
11106
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
11108
Qstate=Q<2>?
No
Yes
11110
PC -> Program Memory Address
Precharge Sense Amp
11112
Qstate=Q<3>?
No
Yes
11114
Read Program Memory
11116
Qstate=Q<4>?
No
Yes
11118
Read Program Memory
Program Memory (Instruction) -> ROMLATCH
Increment PC
11120
Execute1

Figure 111

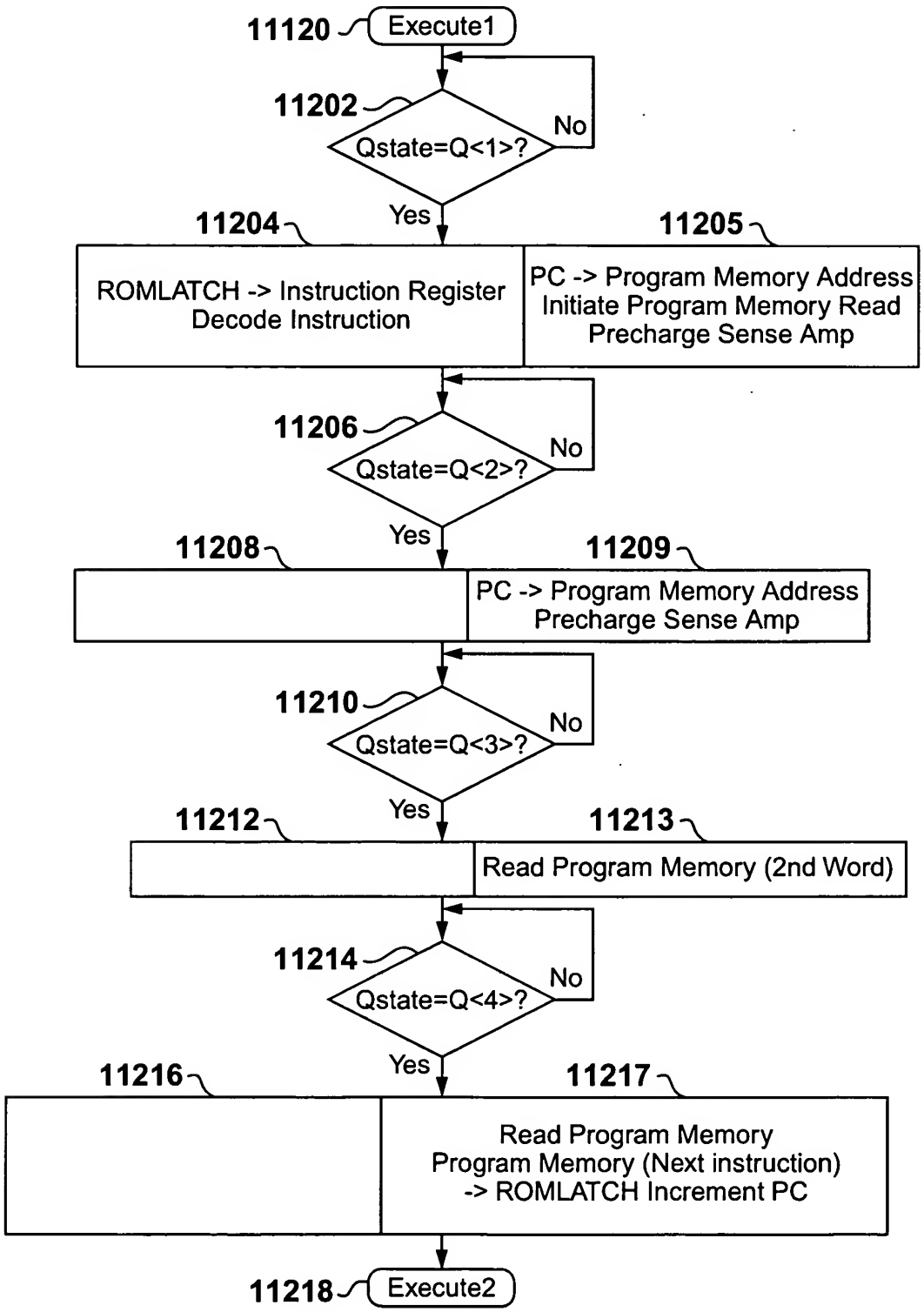

11120
Execute1
11202
Qstate=Q<1>?
No
Yes
11204
11205
ROMLATCH -> Instruction Register
Decode Instruction
PC -> Program Memory Address
Initiate Program Memory Read
Precharge Sense Amp
11206
Qstate=Q<2>?
No
Yes
11208
11209
PC -> Program Memory Address
Precharge Sense Amp
11210
Qstate=Q<3>?
No
Yes
11212
11213
Read Program Memory (2nd Word)
11214
Qstate=Q<4>?
No
Yes
11216
11217
Read Program Memory
Program Memory (Next instruction)
-> ROMLATCH Increment PC
11218
Execute2

Figure 112

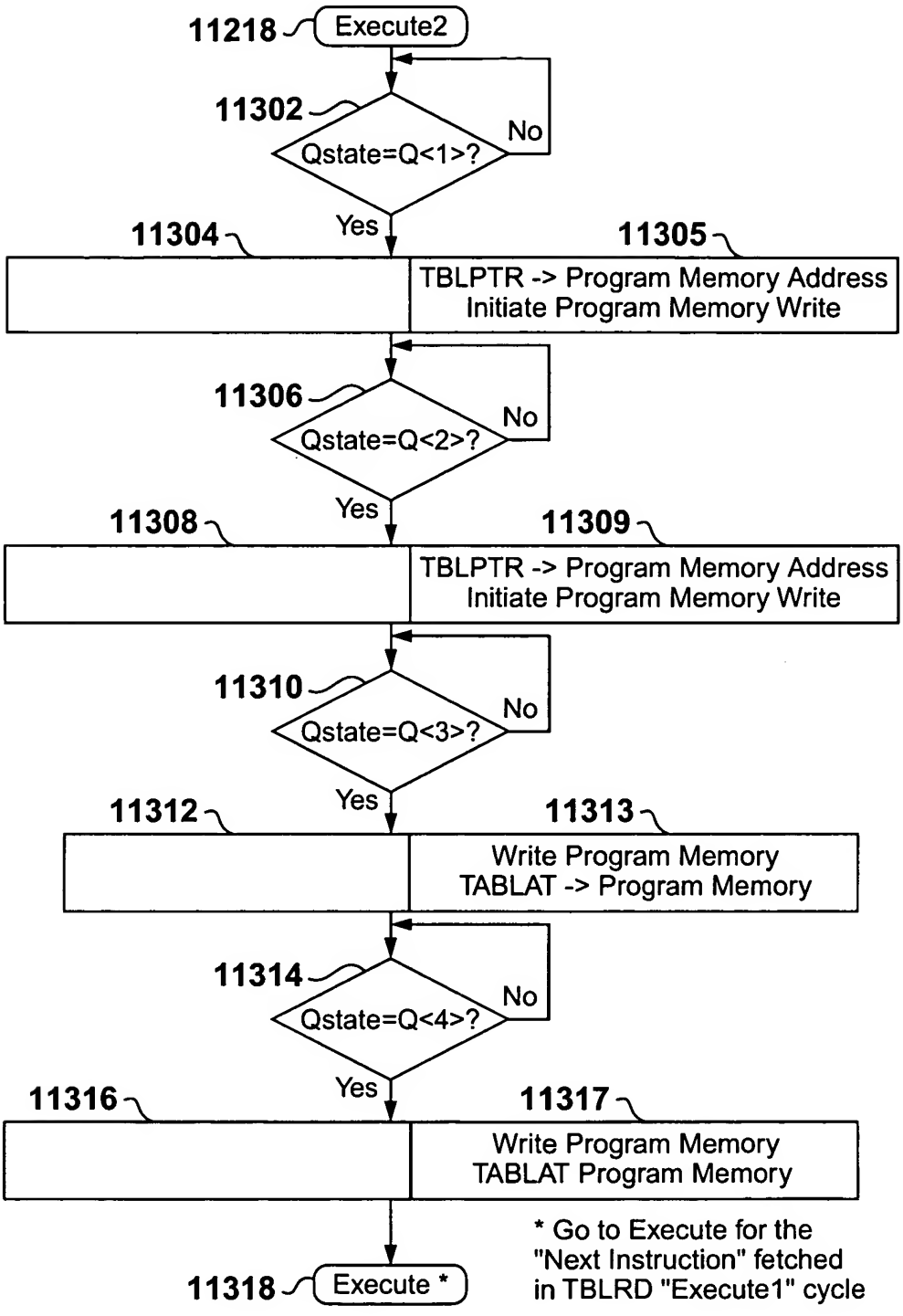

11218
Execute2
11302
Qstate=Q<1>?
No
Yes
11304
11305
TBLPTR -> Program Memory Address
Initiate Program Memory Write
11306
Qstate=Q<2>?
No
Yes
11308
11309
TBLPTR -> Program Memory Address
Initiate Program Memory Write
11310
Qstate=Q<3>?
No
Yes
11312
11313
Write Program Memory
TABLAT -> Program Memory
11314
Qstate=Q<4>?
No
Yes
11316
11317
Write Program Memory
TABLAT Program Memory
11318
Execute *
* Go to Execute for the
"Next Instruction" fetched
in TBLRD "Execute1" cycle

Figure 113

Opcode <11:8>

Opcode <15:12>

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	***	MOVLB	MULWF		DECF				SUBLW	IORLW	XORLW	ANDLW	RETLW	MULLW	MOVLW	ADDLW
1	IORWF				ANDWF				XORWF				COMF			
2	ADDWFC				ADDWF				INCF				DECFSZ			
3	RRCF				RLCF				SWAPF				INCFSZ			
4	RRNCF				RLNCF				INFSNZ				DCFSNZ			
5	MOVF				SUBFWB				SUBWFB				SUBWF			
6	CPFSLT		CPFSEQ		CPFSGT		TSTFSZ		SETF		CLRF		NEGF		MOVWF	
7	BTG															
8	BSF															
9	BCF															
A	BTFSS															
B	BTFSC															
C	MOVFF															
D	BRA								RCALL							
E	BZ	BNZ	BC	BNC	BV	BNV	BN	BNN	open				CALL		LFSR	GOTO
F	NOP (2ND WORD)															

***Special Instructions

0000	NOP
0001	HALT (NOTE: Emulation mode only.)
0003	SLEEP
0004	CLRWDT
0005	PUSH
0006	POP
0007	DAW
0008	TBLRD *
0009	TBLRD *+
000A	TBLRD *-
000B	TBLRD +*
000C	TBLWT *
000D	TBLWT *+
000E	TBLWT *-
000F	TBLWT +*
0010, 0011	RETFIE
0012, 0013	RETURN
00E0	TRAP
00E1	TRET
00FF	RESET

Figure 114

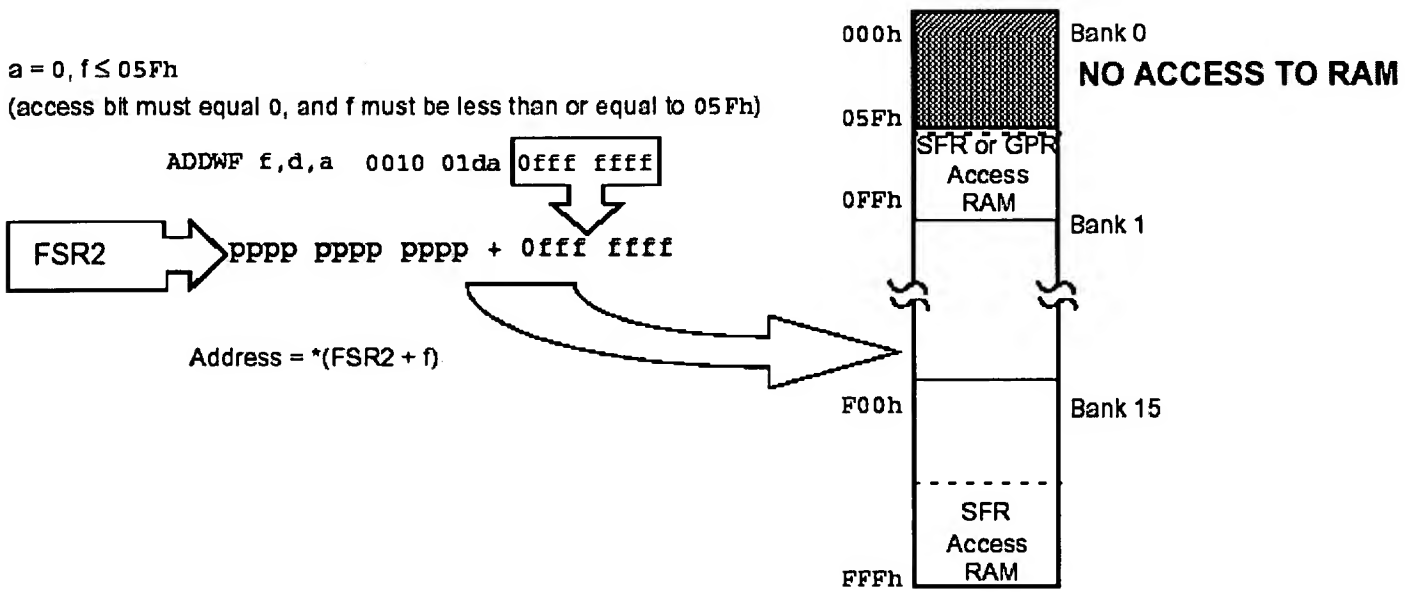

a = 0, f ≤ 05Fh
(access bit must equal 0, and f must be less than or equal to 05Fh)
ADDWF f,d,a 0010 01da 0fff ffff
FSR2
pppp pppp pppp + 0fff ffff
Address = *(FSR2 + f)
000h
Bank 0
NO ACCESS TO RAM
05Fh
SFR or GPR
Access
RAM
0FFh
Bank 1
F00h
Bank 15
SFR
Access
RAM
FFFh

Figure 115

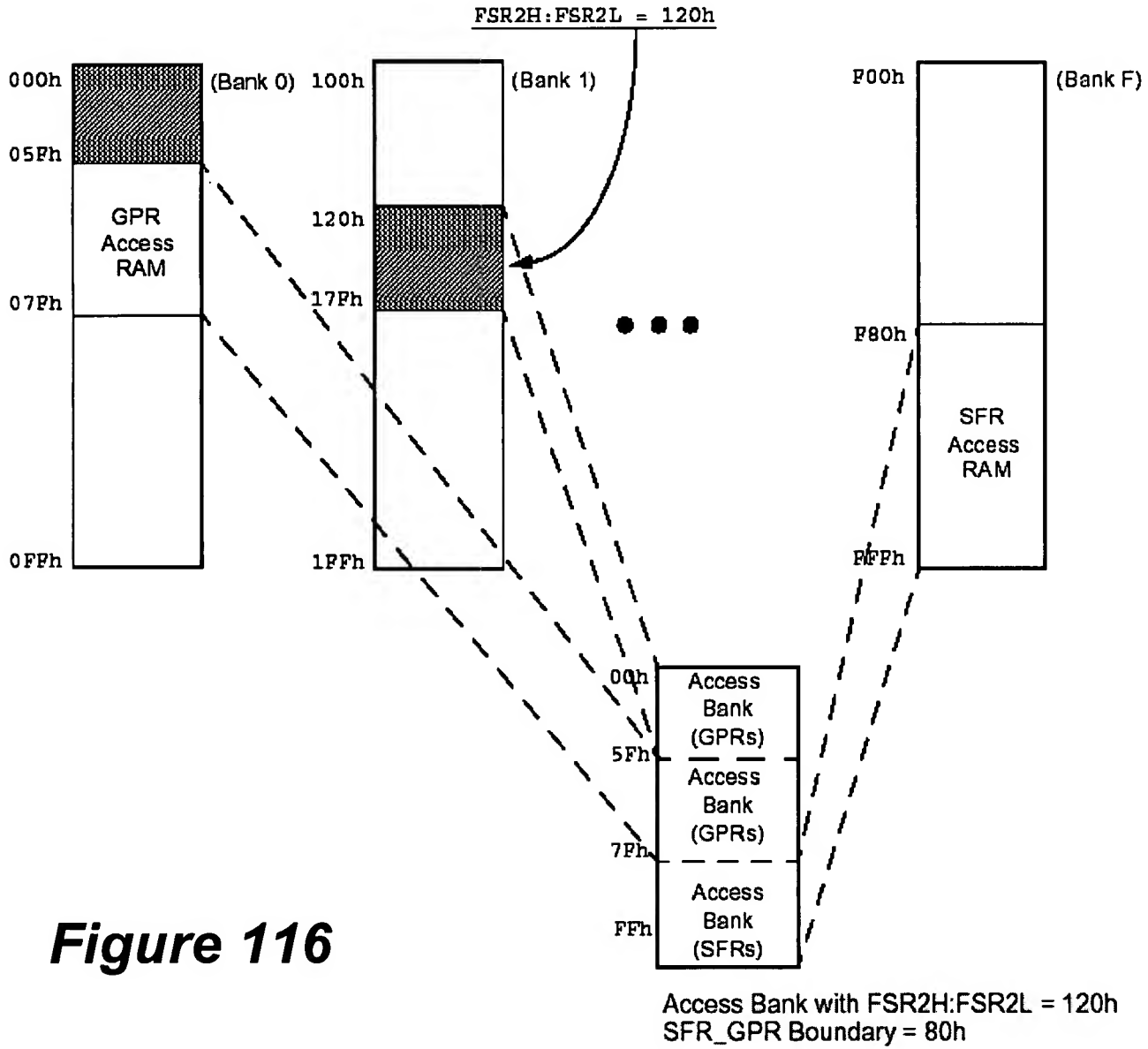


Figure 116

Access Bank with FSR2H:FSR2L = 120h
SFR_GPR Boundary = 80h